INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Sistema de Recomendação de VideoJogos

Rosária Patrícia Firmino Bunga

Mestrado em Engenharia de Telecomunicações e Informática

Orientador:
Doutor Ricardo Daniel Santos Faro Marques Ribeiro, Professor Associado,
Iscte – Instituto Universitário de Lisboa

Co-Orientador:
Doutor Fernando Manuel Marques Batista, Professor Associado,
Iscte – Instituto Universitário de Lisboa

Novembro, 2020

TECNOLOGIAS
E ARQUITETURA

Sistema de Recomendação de Videojogos

Rosária Patrícia Firmino Bunga

Mestrado em Engenharia de Telecomunicações e Informática

Orientador:
Doutor Ricardo Daniel Santos Faro Marques Ribeiro, Professor Associado,
Iscte – Instituto Universitário de Lisboa

Co-Orientador:
Doutor Fernando Manuel Marques Batista, Professor Associado,
Iscte – Instituto Universitário de Lisboa

Novembro, 2020

Agradecimentos

Agradeço aos meus orientadores, Professor Fernando Batista e Professor Ricardo Ribeiro, por me terem direcionado ao longo dessa dissertação, por toda a dedicação, tempo, experiência, paciência, ideias, opiniões e orientação.

À minha família, o meu agradecimento pela confiança, pelo apoio e pelas condições proporcionadas durante toda a minha jornada acadêmica.

Aos meus amigos e ao Ednovan Pedro, obrigada pelo apoio, compreensão, suporte e muita motivação.

A todos os envolvidos nessa trajetória o meu sincero “Obrigada”.

Resumo

Esta dissertação, foca-se no estudo e comparação do desempenho de algoritmos de recomendação baseados em filtragem colaborativa, com o objetivo de propor um sistema de recomendação de videojogos. Esse sistema utiliza informações provenientes da plataforma Steam, que podem ser descritos como dados implícitos, e que posteriormente foram transformados em classificações explícitas para serem usadas nos algoritmos. Os algoritmos foram implementados com recurso à biblioteca *Surprise*, que permite criar e avaliar sistemas de recomendação baseados em dados explícitos. O trabalho foca-se em abordagens computacionalmente menos exigentes, demostrando que as mesmas podem obter bons resultados. Os algoritmos são avaliados e comparados entre si usando métricas como RSME, MAE, Precision@k, Recall@k e F1@k.

Palavras-Chave: Sistema de Recomendação; Filtragem Colaborativa; Classificação Implícita; Validação Cruzada; Avaliação Offline; Videojogos.

Abstract

This dissertation focuses on the study and compare of the performance of collaborative filtering algorithms, with the intent of proposing a videogame-oriented recommendation system. This system uses information from the video game platform "*Steam*", which can be described as implicit feedback, and that were later transformed into explicit feedback. These algorithms were implemented using Python's *Surprise* library, that allows to create and evaluate recommender systems that deal with explicit data. The work focuses on computationally fewer demanding approaches, demonstrating that they can obtain good results. The algorithms are evaluated and compared with each other using metrics such as RSME, MAE, Precision@k, Recall@k and F1@k.

Keywords: Recommendation System; Collaborative Filtering; Implicit Feedback; Cross-Validation, Offline Evaluation; Video Games.

Índice

Índice de Quadros

Índice de Figuras

Lista de Abreviaturas e Siglas

CBF – *Contend Based Filtering*

CF – *Collaborative Filtering*

KNN – *K-Nearest Neighbors*

MAE – *Mean Average Error*

NMF – *Non-Negative Matrix Factorization*

MF – *Matrix Factorization*

RMSE – *Root Mean Squared Error*

RS – *Recommendation Systems*

SGD – *Stochastic Gradient Descent*

SVD – *Singular Value Decomposition*

SVD++ – *Singular Value Decomposition Plus Plus*

Capítulo 1 – Introdução

Este capítulo apresenta um enquadramento do tema da dissertação, a motivação para o desenvolvimento do tema, bem como as questões e objetivos de investigação e a estrutura organizacional dessa dissertação.

1.1. Enquadramento do Tema

Nos últimos anos as pessoas têm vindo a utilizar cada vez mais a Internet. E, em simultâneo, tem-se verificado um rápido crescimento do conteúdo, produto ou serviço fornecido por sites como Google, Youtube, Amazon, Netflix, Steam, entre outros.

A grande diversidade de informação disponível na Internet, começou facilmente a sobrecarregar os utilizadores, deixando-os indecisos e assim dificultando o processo de tomada de decisão. Essa grande quantidade de informação, em vez de gerar um benefício, passou a ser um problema para os utilizadores. Compreendeu-se que, embora a escolha seja boa, mais escolha nem sempre é melhor [1].

Então, à medida em que esse fenómeno se foi intensificando, surgiu uma necessidade de filtrar a informação relevante de toda uma gama de alternativas disponíveis, para um determinado utilizador, de modo a facilitar a escolha de produtos ou serviços mais adequados para si.

Para minimizar ou resolver esse problema de filtragem da informação surgiram os sistemas de recomendação, que passaram a auxiliar no processo de tomada de decisão, fornecendo recomendações personalizadas para os utilizadores [2].

A recomendação é algo com que estamos todos familiarizados, quer um amigo nos recomende um novo livro para ler ou um filme para ver, tudo se resume a dar boas opções e ajudar a fazer uma escolha. Essas recomendações geralmente são dadas com base no conhecimento sobre aquilo de que gostamos. Os sistemas de recomendação funcionam exatamente dessa maneira, na medida em que o sistema tenta usar o histórico do utilizador ou o seu perfil para prever quais produtos ou serviços a recomendar.

Todas as plataformas mencionadas acima passaram a implementar algum tipo de sistema de recomendação que atende às necessidades de seus utilizadores, fornecendo conteúdo personalizado com o objetivo de proporcionar ao utilizador uma melhor

experiência, aumentando desta forma a fidelidade do utilizador, vendendo mais e diversos produtos e finalmente, melhorando as receitas dessas empresas [1].

Tendo isso em consideração, esse estudo foca-se no domínio dos videojogos, devido ao crescimento exponencial desse mercado e às suas particularidades. Utilizamos, especificamente a plataforma de videojogos Steam. Segundo o site do Steam, atualmente têm mais de 30.000 jogos diferentes para os sistemas operativos Windows, Mac e Linux.

Num mercado como esse, de rápida evolução, que tem uma grande quantidade de jogos clássicos, novos lançamentos e jogos *indie*, e que a cada mês vão surgindo dezenas de jogos, os utilizadores/jogadores revelam ter muitas dificuldades para encontrar novos jogos em que possam estar interessados. Ter um jogo recomendado com base nas suas preferências pessoais pode ajudar esses utilizadores a comprar um determinado jogo.

1.2. Motivação e Relevância do Tema

Atualmente, a indústria de videojogos lidera a indústria do entretenimento com receita, em 2019, de $120,1 mil milhões a nível global, com $64,4 mil milhões em dispositivos móveis, $29,6 mil milhões em PCs e $15,4 mil milhões jogos de consola[1]. É uma indústria altamente lucrativa. Segundo a Valve Corporation[2], o Steam é o maior distribuidor mundial de jogos para PC, ocupando 75% do mercado global, hoje em dia. Em 2019, havia cerca de 95 milhões de utilizadores ativos no Steam. O maior número de utilizadores simultâneos do Steam é 24,5 milhões, observado em abril de 2020. E todos os anos são lançados novos jogos, só em 2019 o Steam lançou 8290 jogos.

Com um mercado tão diversificado como esse existe nitidamente a necessidade de se implementarem sistemas de recomendação com alta precisão que forneçam aos utilizadores jogos relevantes desconhecidos e/ou novos lançamentos que atendam aos seus gostos. Isto, torna-se ainda mais evidente, quando olhamos os registos do Steam de 2014, onde se verifica que cerca de 37% dos jogos comprados nunca foram jogados pelos utilizadores que os compraram.

Um utilizador joga os seus jogos preferidos muitas vezes, mas também quer descobrir novos jogos que lhe sejam relevantes. Isso apresenta-se como forma de desafio para esse

[1] SuperData, '2019 Year in Review: Digital games and interactive media', SuperData Research Holdings Inc., 2019, p. 8
[2] https://steamcommunity.com/groups/steamworks/announcements/detail/1697229969000435735

mercado: a necessidade de videojogos que incentivem o utilizador a voltar e ajudar os utilizadores a encontrar novos jogos que serão consumidos tanto quanto os que já gostaram.

Os sistemas de recomendação podem beneficiar muito o mercado de videojogos, pela sua capacidade de sugerir novos jogos aos utilizadores de forma personalizada. Apesar disso, há ainda muito a explorar sobre os sistemas de recomendação no domínio dos videojogos.

1.3. Questões e Objetivos de Investigação

Com base nas observações acima, o objetivo deste trabalho é desenvolver um sistema de recomendação que fornece sugestões de videojogos a um utilizador com base no seu histórico de jogos e gostos de utilizadores semelhantes a ele. Faremos a implementação de várias abordagens baseadas em diferentes algoritmos de filtragem colaborativa, utilizando para isso os dados da plataforma Steam, recorrendo a dados implícitos para inferir as classificações explícitas dos utilizadores, de modo a validar a sua adequação ao domínio dos videojogos.

Os objetivos específicos dessa dissertação são os seguintes:

- Estudo e apresentação do estado da arte dos sistemas de recomendação focando essencialmente o domínio dos videojogos;
- Transformação de dados implícitos em classificações explícitas que posteriormente serão utilizadas nos algoritmos de filtragem colaborativa;
- Implementação de várias abordagens de recomendação baseadas em diferentes algoritmos de filtragem colaborativa;
- Análise dos resultados obtidos, utilizando as métricas de avaliação selecionadas.

Pretendemos responder às seguintes questões de investigação com referência ao domínio dos videojogos:

- Qual é o desempenho dos algoritmos de recomendação em comparação entre si, quando aplicámos sobre dados implícitos?

- Considerando que o tempo de jogo de um utilizador é um dado implícito do sistema, o mesmo pode servir como uma boa representação das preferências do utilizador?

1.4. Estrutura e Organização da Dissertação

O presente trabalho está organizado em cinco capítulos:

O primeiro capítulo introduz o tema da investigação e objetivos da dissertação, bem como uma breve descrição da estrutura da dissertação.

O segundo capítulo aborda o contexto teórico, contém a revisão bibliográfica dos temas abordados nesta dissertação, com breve explicação de cada tópico de modo a facilitar a compreensão dos temas tratados ao longo da dissertação.

O terceiro capítulo é dedicado à exploração e tratamento do conjunto de dados, bem como dos algoritmos e métricas de avaliação utilizados.

O quarto capítulo apresenta a análise dos resultados obtidos, de acordo com a abordagem que se entendeu apropriada.

No quinto e último capítulo apresentam-se as conclusões finais deste trabalho bem como as recomendações, limitações e trabalhos futuros que permitem a continuidade deste estudo.

Capítulo 2 – Conceitos e Trabalhos Relacionados

Este capítulo, esclarece os conceitos teóricos relacionados com o tema de sistemas de recomendação e analisa os trabalhos realizados nessa área. Na seção 2.1 introduzimos o conceito de sistemas de recomendação e técnicas de filtragem de recomendação. Na seção 2.2 apresentamos uma visão geral sobre tipos de classificações. Na seção 2.3 expomos os métodos de avaliação. Na seção 2.4 descrevemos alguns dos desafios dos sistemas de recomendação. Finalmente, na seção 2.5, analisamos os trabalhos realizados sobre os sistemas de recomendação no domínio dos videojogos.

2.1. Sistemas de Recomendação

Os Sistemas de Recomendação ou *Recommendation Systems* (RS), são ferramentas e técnicas que fornecem sugestões de itens para serem usados pelos utilizadores, preveem preferências e recomendam itens para os utilizadores de acordo com as suas preferências, em uma ampla variedade de domínios [1]. Esses sistemas auxiliam no processo da tomada de decisão fornecendo recomendações personalizadas para seus utilizadores.

As técnicas de filtragem de recomendação são classificadas de acordo com a abordagem utilizada para fazer a previsão e fornecer as informações relevantes ao utilizador. Na literatura, existem tradicionalmente três abordagens, a filtragem colaborativa ou *Collaborative Filtering* (CF) que utiliza as interações utilizador-item ou o comportamento do utilizador como classificações; a filtragem baseada em conteúdo ou *Contend-Based Filtering* (CBF) que utiliza as descrições de itens e descrições de utilizadores, como perfis textuais ou palavras-chaves relevantes; e, a filtragem híbrida ou *Hybrid Filtering* que combina diferentes técnicas para fornecer uma melhor recomendação [3], [4], [5].

2.1.1. Filtragem Colaborativa

A filtragem colaborativa analisa os dados históricos de classificação do utilizador e, de acordo com esses relacionamentos, essa técnica pode obter os utilizadores semelhantes, utilizando alguns algoritmos de similaridade, e prever quais itens têm relevância para o utilizador [6]. Supõe-se que as opiniões de outros utilizadores podem ser selecionadas e agregadas de forma a fornecer uma previsão razoável da preferência

do utilizador ativo. Essa técnica presume que, se os utilizadores concordarem sobre a qualidade ou relevância de alguns itens, eles provavelmente concordarão sobre outros itens [7]. A filtragem colaborativa depende da relação entre utilizadores e itens, que normalmente são representados em uma matriz de classificação com cada elemento representando uma classificação específica do utilizador em um item específico.

Existem dois tipos de métodos comumente utilizados na filtragem colaborativa, que são referidos como métodos baseados em memória ou *memory-based methods* e métodos baseados em modelo ou *model-based methods* [3], [5].

Métodos baseados em memória são também chamados de algoritmos de filtragem colaborativa baseados em vizinhança ou *k-NN collaborative filtering*, nos quais as classificações de interações de utilizador-item são previstas com base em suas vizinhanças. Essas vizinhanças podem ser definidas de duas maneiras:

Filtragem colaborativa baseada no utilizador ou *User-based collaborative filtering*: identifica utilizadores que compartilham interesses semelhantes com o utilizador alvo e calcula a preferência do utilizador alvo. A ideia básica é determinar os utilizadores, que são semelhantes ao utilizador alvo e em seguida, fazer recomendações com base nas preferências desses utilizadores semelhantes. Por exemplo, se Alice e Bob classificaram filmes de maneira semelhante no passado, então pode-se usar as classificações de Alice no filme *Terminator* para prever as classificações de Bob neste filme. Em geral, os *k* utilizadores mais semelhantes a Bob podem ser usados para fazer previsões de classificação para Bob. As funções de similaridade são calculadas entre as linhas da matriz de classificações para descobrir utilizadores semelhantes [1], [3].

Filtragem colaborativa baseada em item ou *Item-based collaborative filtering*: a ideia principal dos algoritmos baseados em itens é calcular previsões usando a similaridade entre os itens e não a similaridade entre os utilizadores. Analisa semelhanças entre itens e usa esses itens semelhantes para identificar o conjunto de itens a serem recomendados. Por exemplo, as classificações de Bob em filmes de ficção científica semelhantes como *Alien* e *Predator* podem ser usadas para prever sua classificação no *Terminator*. Funções de similaridade são calculadas entre as colunas da matriz de avaliações para descobrir itens semelhantes [2], [3].

Métodos baseados em modelos utilizam técnicas de *Machine Learning* e *Data Mining*, pois são usados no contexto de modelos preditivos. Essas técnicas podem recomendar rapidamente um conjunto de itens pelo fato de usarem um modelo pré-computado e provarem produzir resultados de recomendação semelhantes às técnicas de recomendação baseadas na vizinhança. Exemplos dessas técnicas incluem técnica de *Bayesian Clustering*, *Latent Semantic Analysis*, *Singular Value Decomposition* [8], *Dimensionality Reduction* e *Regression* [6], [9]. As técnicas baseadas em modelos analisam a matriz de itens do utilizador para identificar as relações entre os itens. Conseguem resolvem os problemas de dispersão associados aos sistemas de recomendação [5].

2.1.2. Filtragem Baseada no Conteúdo

Um algoritmo de filtragem baseada em conteúdo ou *Contend-Based Filtering* (CBF), utiliza as descrições de itens para criar recursos e atributos para combinar os perfis dos utilizadores. Baseia suas previsões em informações sobre utilizadores e itens individuais e ignoram as contribuições de outros utilizadores, isto é, embora tal abordagem deva contar com informações adicionais sobre itens e preferências do utilizador, ela não requer a existência de uma grande comunidade de utilizadores ou um histórico de classificação [2], [10]. Este tipo de sistema necessita de uma quantidade considerável de informações que descrevam a natureza de cada item. Todas essas informações, coletadas como um vetor de recursos para cada item, são consideradas indispensáveis. A falta dessa informação às vezes dificulta ou impossibilita o uso dessa técnica [11].

2.1.3. Filtragem Híbrida

A filtragem híbrida combina as técnicas de filtragem colaborativa e filtragem baseada em conteúdo de várias maneiras para fazer recomendações [1]. Beneficia das vantagens complementares da combinação de técnicas, visando a melhoria da precisão e eficiência da recomendação. A abordagem híbrida é muitas vezes usada para superar as limitações do sistema de recomendação, como inicialização a frio e problemas de dispersão [10]. Os sistemas híbridos podem ser particularmente benéficos quando os algoritmos envolvidos cobrem diferentes casos de uso ou diferentes aspetos do conjunto de dados. Por exemplo, a filtragem colaborativa baseada em item sofre quando ninguém classificou um item ainda, mas as abordagens baseadas em conteúdo não [12]. Um sistema de recomendação

híbrido pode usar a similaridade do texto de descrição para combinar o novo item com os itens existentes, permitindo que seja recomendado de qualquer maneira e aumentar a influência da filtragem colaborativa conforme os utilizadores avaliam o item, da mesma forma, os utilizadores podem ser definidos pelo conteúdo dos itens de que gostam, bem como pelos próprios itens [3].

2.2. Tipos de Classificação

As classificações em um sistema de filtragem colaborativa podem assumir uma variedade de formas [10]:

- Classificações escalares podem consistir em classificações numéricas, como as de 1 a 5 estrelas;
- Classificações ordinais, como concordo totalmente, concordo, neutro, discordo, discordo totalmente;
- Classificações binárias entre concordo / discordo ou bom / mau;
- Classificações unárias podem indicar que um utilizador observou ou comprou um item ou, de outra forma, classificou o item positivamente. A ausência de uma classificação indica que não temos informações relacionadas ao utilizador com o item.

As classificações podem ser obtidas por meios explícitos, meios implícitos ou ambos. As classificações explícitas são aquelas em que um utilizador é solicitado a fornecer uma opinião sobre um item. As classificações implícitas são aquelas inferidas das ações de um utilizador.

Para desenvolver um sistema de recomendação baseado em filtragem colaborativa, as preferências dos utilizadores precisam ser aprendidas, para gerar um perfil/modelo de utilizador para a tarefa de previsão [13]. O sucesso de qualquer sistema de recomendação depende em grande parte da sua capacidade de representar as preferências atuais do utilizador. Perfis de utilizadores precisos são indispensáveis para obter recomendações relevantes [14].

Uma abordagem comum para a construção do perfil do utilizador é a obtenção da classificação explícita ou implícita do utilizador. A classificação explícita, como escalas de avaliação, fornece aos utilizadores um mecanismo para expressar inequivocamente os seus interesses nos itens. Por outro lado, a classificação implícita é gerada pelo próprio

sistema de recomendação, inferindo as preferências do utilizador por meio da observação do comportamento do utilizador. O que constitui classificação implícita depende do domínio da aplicação. Normalmente, será um ou vários parâmetros observáveis e mensuráveis que surgem das interações do utilizador com o sistema.

A classificação explícita tem a vantagem de simplicidade, embora a adoção de escalas numéricas/simbólicas aumente a carga cognitiva do utilizador e pode não ser adequada para captar o sentimento do utilizador sobre os itens. Tornando-se difícil obter classificações suficientes e representativas de uma comunidade de utilizadores. Os métodos de classificação implícita são baseados na atribuição de uma pontuação de relevância para ações específicas do utilizador em um item, como salvar, descartar, imprimir, marcar como favorito, etc. A principal vantagem é que eles não exigem um envolvimento direto do utilizador [1]. Existe, também, a possibilidade de se combinar essas duas classificações, gerando as classificações híbridas.

2.3. Metodologia de Avaliação

Os sistemas de recomendação podem ser avaliados usando métodos online ou métodos offline. Em um sistema online, as reações do utilizador são medidas em relação às recomendações apresentadas. Portanto, a participação do utilizador é essencial em sistemas online. Por exemplo, em uma avaliação online de um sistema de recomendação de notícias, pode-se medir a taxa de conversão de utilizadores que clicam em artigos recomendados. Esses métodos de teste são chamados de teste A / B e medem o impacto direto do sistema de recomendação no utilizador final. No final do dia, aumentar a taxa de conversão de itens lucrativos é a meta mais importante de um sistema de recomendação e pode fornecer uma medida real da eficácia do sistema. No entanto, como as avaliações online exigem a participação ativa do utilizador, muitas vezes não é viável usá-las em pesquisa. Nesses casos, são utilizadas avaliações offline com conjuntos de dados históricos. Os métodos offline são, de longe, os métodos mais comuns para avaliar os sistemas de recomendação de uma perspetiva de pesquisa e prática [3].

A estrutura básica para avaliação offline é baseada na configuração treinamento-teste. Começa com um conjunto de dados, que geralmente consiste em uma coleção de classificações ou históricos de utilizadores e, possivelmente, contém informações adicionais sobre utilizadores e/ou itens. Os utilizadores neste conjunto de dados são divididos em dois grupos: o conjunto de treinamento e o conjunto de teste. Um modelo

de recomendação é construído em relação ao conjunto de treinamento [15]. Os utilizadores no conjunto de teste são considerados separadamente e têm suas classificações divididas em duas partes, o conjunto de consultas e o conjunto de destino. O sistema de recomendação recebe o conjunto de consultas como um histórico do utilizador e é solicitado a recomendar itens ou prever classificações para os itens no conjunto de destino; ele é então avaliado em quão bem suas recomendações ou previsões correspondem àquelas apresentadas na consulta. Todo esse processo é frequentemente repetido como na validação cruzada *k-fold* dividindo os utilizadores em *k* conjuntos iguais e usando cada conjunto como o conjunto de teste com a união de todos os outros conjuntos como o conjunto de treinamento, como é possível ver na Figura 1. Os resultados de cada execução podem então ser agregados para avaliar o desempenho geral do recomendador, mitigando os efeitos da variação do conjunto de teste [16], [7].

Figura 1 - Validação cruzada k-fold, para k=5

2.4. Desafios dos Sistemas de Recomendação

Identificámos e descrevemos os desafios dos sistemas de recomendação. Embora não se abordem todos os desafios exaustivamente, concentramo-nos em alguns dos mais relevantes: inicialização a frio, escalabilidade, tendência de popularidade.

Problema de inicialização a frio (*Cold Start Problem)*

O problema de inicialização a frio descreve o fraco desempenho do sistema em lidar com novos itens e novos utilizadores. A inicialização a frio pode ser considerada um subproblema de cobertura, pois mede a cobertura do sistema em um conjunto específico

de itens e utilizadores. O problema fica ainda mais crítico quando a matriz de itens do utilizador aumenta de tamanho, o que significa que quanto maior a dispersão da matriz, menor a cobertura de classificação. Entretanto, já foram propostas abordagens para lidar com esse problema. As abordagens passam pela utilização das técnicas baseadas em conteúdo, técnicas híbridas [1].

Escalabilidade (*Scalability)*

Como os sistemas de recomendação são projetados para ajudar os utilizadores a navegar em grandes coleções de itens, um dos objetivos é escalar para conjuntos de dados reais. Como tal, é frequente o caso que algoritmos negociam outras propriedades, como *precision* ou *recall*, para fornecer resultados rápidos, mesmo para grandes conjuntos de dados consistindo em milhões de itens. Com o crescimento do conjunto de dados, muitos algoritmos ficam mais lentos ou requerem recursos adicionais, como capacidade de computação ou memória. A escalabilidade é normalmente medida experimentando conjuntos de dados crescentes, mostrando como a velocidade e o consumo de recursos se comportam conforme a tarefa aumenta. É importante medir os compromissos impostos pela escalabilidade. Como em muitos casos se espera que os sistemas de recomendação forneçam recomendações rápidas online, também é importante medir a rapidez com que o sistema fornece recomendações[10].

Tendência de Popularidade (*Popularity Bias)*

A tendência de popularidade resulta da significativa preponderância que as técnicas de filtragem colaborativa dão a itens populares (aqueles com mais contagens ou classificações de jogo) sobre outros itens de "*long-tailed*" que podem ser populares apenas entre pequenos grupos de utilizadores. Embora na maioria dos casos recomendar um jogo popular possa resultar em uma boa recomendação, entregar apenas jogos populares não irá aprimorar a experiência geral do utilizador, ignorando interesses mais específicos que os utilizadores possam ter. Isso também significa que o modelo pode ser tendencioso para jogos mais famosos, colocando em desvantagem os outros menos populares [17].

2.5. Trabalhos Relacionados

Analisamos os trabalhos relacionados no domínio dos videojogos, verificando as técnicas que os sistemas propostos utilizam.

Pérez-Marco *et al*. [11] apresentam um sistema híbrido de recomendação de videojogos, por meio do uso da filtragem colaborativa e da filtragem baseada em conteúdo, além da construção de gráficos de relacionamento. O sistema tem como entrada uma lista de classificações de um utilizador, das comunidades dos jogos que o sistema contém e das comunidades dos utilizadores que o sistema contém. É primeiro aplicada a filtragem baseada em conteúdo, para cada item *i* do utilizador ativo, o sistema procura os jogos que estão na comunidade mais próxima desse item. Como resultado, é obtida uma lista com os jogos mais semelhantes aos do utilizador ativo. Em seguida, é aplicada a filtragem colaborativa, restrita aos itens obtidos na etapa anterior e pela obtenção das classificações. Por fim, uma matriz é retornada com os jogos recomendados e seus valores previstos. No caso da filtragem baseada em conteúdo, eles escolheram utilizar técnicas gráficas para encontrar jogos semelhantes aos do jogador ativo, reduzindo a carga computacional do sistema, pois as recomendações são feitas em um subconjunto deles.

Wang *et al.* [18] propõem o STEAMer, um sistema de recomendação de videojogos para a plataforma Steam, que utiliza os dados do utilizador do Steam, em conjunto com um modelo de aprendizagem *Deep Autoencoders* (tipo específico de arquitetura de rede neural que tenta forçar a rede a aprender uma representação compactada dos dados de entrada originais) para aprender os dados do jogo e do utilizador obtidos do banco de dados do Steam para gerar recomendações de jogos em potencial.

Cheuque *et al.* [19] mostram modelos de recomendação baseados respetivamente em *Factorization Machines* (FM), *Deep Neural Networks* (DeepNN) e uma resultante da combinação de ambas (DeepFM), escolhidas pelo seu potencial de receção de múltiplas entradas, bem como diferentes tipos de variáveis de entrada. Todos os algoritmos alcançam resultados melhores do que uma *baseline* ALS (*Alternating Least Squares Model*). Apesar de ser um modelo mais simples do que o DeepFM, o DeepNN foi considerado o algoritmo de melhor desempenho, conseguiu explorar melhor as relações utlizador-item e, embora demore mais do que o FM e o DeepFM para obter resultados competitivos, ele atinge resultados consistentes em diferentes conjuntos de dados.

Analisaram, também, o efeito do sentimento extraído diretamente das análises de jogos e concluíram que não é tão relevante para recomendação quanto se poderia esperar.

Anwar *et al.* [20] propõem um sistema de recomendação que utiliza uma técnica de filtragem colaborativa para sugerir videojogos aos utilizadores. O sistema de recomendação foi implementado utilizando a filtragem colaborativa baseada em itens e a correlação de Pearson, para identificar os jogos não classificados, encontrando semelhanças entre os jogos não classificados e os altamente classificados pelo utilizador. Em seguida, o sistema emprega filtragem colaborativa baseada no utilizador e sugere jogos para o utilizador ativo. Contudo, o sistema não foi capaz de fornecer uma melhor precisão no caso de cenário de inicialização a frio.

Bertens *et al*. [21] propõem um sistema que recomenda videojogos aos utilizadores com base em sua experiência e comportamento com os seus itens, isto é, tempo de jogo, frequência de atividade. O modelo de algoritmo é baseado em *Machine Learning*, o modelo é uma combinação de *Extremely Randomized Trees* (ERTs) e *Deep Neural Networks* (DeepNNs). Os autores apresentam dois modelos para prever quais itens os jogadores ficarão mais interessados em comprar nas suas próximas compras. Os resultados mostram que o desempenho de previsão da DeepNN e do ERT é semelhante. No entanto, o modelo ERT produz resultados ligeiramente melhores e escala mais facilmente em um ambiente de produção.

Pathak *et al.* [22] propuseram um sistema de recomendação baseado em modelos de fatores latentes, e em particular *Bayesian Personalized Ranking* (BPR), que é treinado usando classificação implícita (ou seja, compras vs. não compras), e que usa os recursos treinados de um modelo de recomendação de item para aprender classificações personalizadas sobre pacotes (*bundles*). Mostraram que o modelo é robusto para pacotes frios e que novos pacotes podem ser gerados de maneira eficaz por meio de um algoritmo guloso. O seu foco principal é a compatibilidade item-a-grupo. Mostraram um método para gerar e avaliar a recomendação de pacote personalizado na plataforma de videojogo Steam.

Sifa *et al.* [23] apresentam duas abordagens para sistemas de recomendação Top-N, onde é apresentada ao utilizador uma lista de N recomendações, ou seja, uma factoração de matriz e um modelo de vizinhança baseado em utilizador operando em dimensões reduzidas. Ambos os modelos são baseados na análise arquetípica, um método

semelhante à análise de *cluster*, agrupando assim os utilizadores em *k* arquétipos. Os dados utilizados para esta análise são compostos de *feedback* implícito, especificamente informações sobre propriedade do jogo e tempos de jogo, com o objetivo final de recomendar os jogos que têm o maior tempo de jogo previsto. Os autores comparam seus algoritmos com várias *baselines* e um modelo de vizinhança baseado em itens, que eles conseguiram superar.

Capítulo 3 – Metodologia

Neste capítulo, descrevemos o conjunto de dados e abordagens que foram utilizadas neste trabalho. Na seção 3.1 apresentamos o conjunto de dados para análise. Na seção 3.2 abordamos a conversão de dados implícitos em classificações explícitas. E por último, na seção 3.3 expomos a abordagem proposta na implementação do sistema de recomendação, descrevemos os algoritmos a serem implementados e quais métricas usar para avaliar os modelos.

3.1. Conjunto de Dados

Para a implementação do sistema de recomendação mencionado nesse trabalho atuamos sobre um conjunto de dados obtido da coleção partilhada por Julian McAuley[3]. O conjunto de dados consiste no histórico de compras dos utilizadores australianos da plataforma Steam, ordenado por utilizadores, indicando para cada um a lista de itens adquiridos com uma pequena coleção de metadados conforme o tempo de jogo. Expandimos a lista de itens ou o pacote de itens de cada utilizador do conjunto de dados original, de maneira a que cada registo do conjunto pudesse ser visto como uma interação utilizador/item.

A Tabela 1, mostra os atributos do conjunto de dados originado, com um breve resumo sobre cada um deles. Ao observar os atributos do conjunto de dados percebemos que não existe uma classificação que pudesse ser utilizada como medida da preferência de um utilizador por um determinado jogo, ou seja, o nosso conjunto de dados não tem classificação explícita. Por isso, para a implementação dos algoritmos de recomendação recorremos a classificação implícita para inferir classificação explícita.

A Figura 2 mostra um excerto do conjunto de dados.

[3] http://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data

Tabela 1 - Resumo dos atributos do conjunto de dados

Atributo	Tipo de dado	Descrição
user_id	str	Identificador único do utilizador
steam_id	int	Identificador único para cada utilizador do Steam
user_url	str	URL do utilizador
item_id	str	Identificador único do item/jogo
item_name	str	Nome do item/jogo
playtime_2weeks	int	O número total de minutos jogados nas últimas duas semanas
playtime_forever	int	O número total de minutos jogados

	steam_id	item_id	user_id	user_url	item_name	playtime_2weeks	playtime_forever
0	76561197960269200	10	ONIONZZZ	http://steamcommunity.com/id/ONIONZZZ	Counter-Strike	0	15170
1	76561197960269200	10180	ONIONZZZ	http://steamcommunity.com/id/ONIONZZZ	Call of Duty: Modern Warfare 2	0	13
2	76561197960269200	10190	ONIONZZZ	http://steamcommunity.com/id/ONIONZZZ	Call of Duty: Modern Warfare 2 - Multiplayer	0	23
3	76561197960269200	10500	ONIONZZZ	http://steamcommunity.com/id/ONIONZZZ	Empire: Total War	0	722
4	76561197960269200	10680	ONIONZZZ	http://steamcommunity.com/id/ONIONZZZ	Aliens vs. Predator	0	8
...	...	...	...	...	...	...	...
2794721	76561198308665440	6310	76561198308665434	http://steamcommunity.com/profiles/76561198308...	The Longest Journey	42	42
2794722	76561198308665440	72850	76561198308665434	http://steamcommunity.com/profiles/76561198308...	The Elder Scrolls V: Skyrim	0	2842
2794723	76561198308665440	7670	76561198308665434	http://steamcommunity.com/profiles/76561198308...	BioShock	0	219
2794724	76561198308665440	8850	76561198308665434	http://steamcommunity.com/profiles/76561198308...	BioShock 2	0	131
2794725	76561198308665440	8870	76561198308665434	http://steamcommunity.com/profiles/76561198308...	BioShock Infinite	0	438

Figura 2 – Amostra do conjundo de dados (n = 10)

O conjunto de dados foi filtrado: todos os jogadores com menos de 30 jogos foram removidos do conjunto de dados, porque em comparação com os restantes jogadores do conjunto, esses eram muito escassos. Além disso, todos os jogos que foram jogados por menos de dez jogadores também foram removidos e apenas jogos que tiveram um tempo de jogo superior a zero minutos foram considerados. Com a filtragem de dados, obtivemos

um novo conjunto de dados contendo 33.307 utilizadores únicos e 6.387 jogos únicos. As estatísticas descritivas básicas do conjunto de dados são mostradas na Tabela 2.

Tabela 2 - Estatísticas do conjunto de dados

Estatísticas descritivas	
Número de registos	2.794.726
Número de utilizadores únicos	33.307
Número de item/jogos únicos	6.387

3.2. Transformação da Classificação Implícita

Cada sistema de recomendação deve desenvolver e manter um perfil ou modelo de utilizador que contenha as preferências do mesmo. Embora a existência de um perfil de utilizador seja central para todo sistema de recomendação, a maneira como essas informações são adquiridas e exploradas depende da técnica de recomendação.

Os sistemas de recomendação dependem da compreensão das preferências do utilizador para ajustar a resposta e produzir uma saída personalizada. As preferências do utilizador podem ser capturadas de duas formas: como classificações explícitas ou classificações implícitas [1].

Tal como descrito na seção 3.1, o conjunto de dados utilizado não fornece as classificações explícitas dos utilizadores. Para os nossos algoritmos de filtragem colaborativa, precisamos de classificações do utilizador para a recomendação de itens. A maioria das abordagens para entender as preferências ou gostos do utilizador baseia-se em ter classificações explícitas dos utilizadores. No entanto, em muitas situações da vida real, é preciso confiar nas classificações implícitas, como quantas vezes um utilizador ouviu uma música ou jogou um jogo. Considerando a importância referida em pesquisas anteriores sobre a relação entre classificações explícita e implícita em sistemas de recomendação [24], [25], escolhemos o tempo de jogo total, *"playtime_forever"*, para inferirmos as classificações explícitas do utilizador de modo a compreendermos as preferências do mesmo.

Embora as classificações implícitas possam ser coletadas constantemente e não exijam esforços adicionais por parte do utilizador ao inferir as preferências do mesmo a partir do

seu comportamento com o sistema, não se pode ter certeza, se interpretamos corretamente o comportamento do utilizador. Ainda assim, em [26] os autores relatam que em alguns domínios, como estações de rádio online personalizadas, a coleta de classificações implícitas pode até mesmo resultar em perfis de utilizadores mais precisos do que pode ser feito com classificações explícitas. Além disso, foi também discutido que os dados de preferência implícitos podem na realidade ser mais objetivos, uma vez que não há preconceito decorrente de utilizadores respondendo de uma forma socialmente desejável [27] e não há problemas de autoimagem ou qualquer necessidade de manter uma imagem para outros. Isso enfatiza que a adequada interpretação da classificação implícita pode ser altamente dependente do respetivo domínio.

Observamos o tempo de jogo total, em minutos, que um utilizador jogou um determinado jogo, *playtime_forever*, explorando a ideia de usar o tempo de jogo para quantificar a possibilidade de um item ser relevante para um utilizador específico. Pois, tem-se como premissa que se um utilizador joga um jogo por muito tempo, podemos supor que ele gosta desse jogo. Para a tarefa de conversão da classificação implícita em classificação explícita, recorremos a algumas funções da biblioteca Pandas, do Python. Aplicamos a função *cut()* para agrupar os tempos de jogos em cinco intervalos. Depois aplicamos a função *rank()* para atribuir uma classificação, valores iguais são atribuídos a uma classificação que é a média das classificações desses valores. O nosso sistema de classificação tornou-se uma escala de classificação de 1 a 5. A Figura 3, contém um excerto do resultado final.

	Steam id	item_id	Item name	Playtime 2weeks	Playtime forever	rating
2794716	76561198308665400	45770	Dead Rising 2: Off the Record	0	72	3
2794717	76561198308665400	46510	Syberia 2	0	40	2
2794718	76561198308665400	466500	35MM	0	19	1
2794719	76561198308665400	55230	Saints Row: The Third	0	1029	5
2794720	76561198308665400	6300	Dreamfall: The Longest Journey	57	68	2
2794721	76561198308665400	6310	The Longest Journey	42	42	2
2794722	76561198308665400	72850	The Elder Scrolls V: Skyrim	0	2842	5
2794723	76561198308665400	7670	BioShock	0	219	4
2794724	76561198308665400	8850	BioShock 2	0	131	3
2794725	76561198308665400	8870	BioShock Infinite	0	438	4

Figura 3 – Excerto do conjunto de dados final, contendo apenas a informação mais relevante.

Na Figura 4 temos as classificações do conjunto de dados distribuídas em intervalos de 1 a 5: é possível verificar que a distribuição das classificações é quase uniforme.

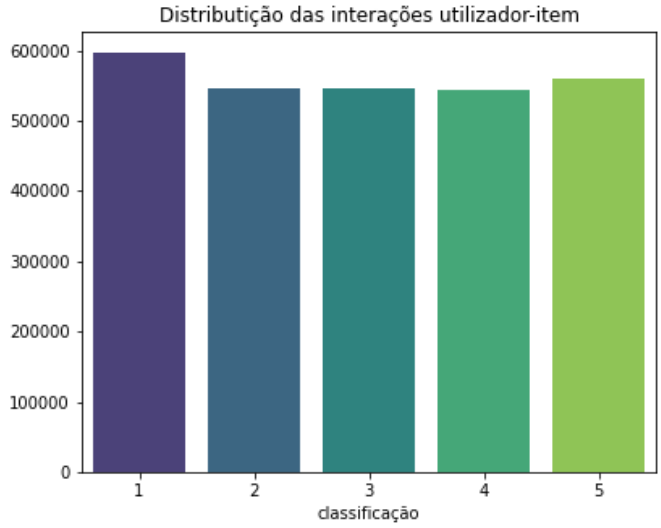

Figura 4 - Distribuição das interações utilizador item pelas classificações

3.3. Abordagem de Recomendação Baseada em Filtragem Colaborativa

Conforme discutido anteriormente, existem três métodos comuns para recomendações na literatura, filtragem baseada no conteúdo, filtragem colaborativa e filtragem híbrida. Neste trabalho, vamos concentrar-nos no método de filtragem colaborativa. Na filtragem colaborativa, recomendações para cada utilizador são geradas fazendo comparações com o gosto por uma alternativa em relação a outros utilizadores que qualificaram o produto de forma semelhante ao utilizador ativo [8].

Como vimos na seção 2.3, uma abordagem comum nos sistemas de recomendação é dividir o conjunto de dados em dois conjuntos: o conjunto de treinamento e o conjunto de teste. Um modelo de recomendação é construído em relação ao conjunto de treinamento. E o conjunto de teste por sua vez é subdividido em dois, o conjunto de consultas e o conjunto de destino. O recomendador receberá o conjunto de treinamento e o conjunto de consultas e é solicitado a sugerir itens ou prever classificações para os itens no conjunto de destino.

Para a implementação dos vários algoritmos de recomendação, trabalhamos com a biblioteca Surprise[4] – Simple Python Recommendation System, construída por Nicolas

[4] https://surprise.readthedocs.io/en/stable/

Hug. Essa biblioteca fornece diversos algoritmos de recomendação e ferramentas para avaliar, analisar e comparar algoritmos. Em seguida, encontram-se os algoritmos de recomendação selecionados para implementação.

Algoritmos baseados no Vizinho Mais Próximo (*Nearest Neighbors*)

A ideia primária dos algoritmos baseados em vizinhança é usar a similaridade utilizador-utilizador ou similaridade item-item para fazer recomendações a partir de uma matriz de classificações [3].

KNNBasic: É um algoritmo de filtragem colaborativa baseada na vizinhança. O conceito de vizinhança implica que precisamos determinar utilizadores semelhantes ou itens semelhantes para fazer previsões [3]. A previsão $\hat{r}_{ui}$ é definida como

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u,v).r_{vi}}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{3.1}$$

onde $sim(u,v)$ é a similaridade entre vetores de classificação de dois utilizadores *u* e *v*. E pode ser calculada com base na similaridade do cosseno, com base no coeficiente de correlação de Pearson ou com base na similaridade da diferença média quadrática. A notação de "^" no topo de r_{ui} indica uma classificação prevista, ao contrário de uma que já foi observada na matriz de classificações original. $N_i^k(u)$ representa o conjunto de *k* utilizadores mais próximos do utilizador alvo *u*, que especificou classificações para o item *i* [3].

KNNWithMeans: Este algoritmo de filtragem colaborativa, é idêntico ao anterior, mas tem em consideração as avaliações médias de cada utilizador. A média ponderada da classificação centrada na média de um item no grupo de pares top-*k* do utilizador alvo *u* é usada para fornecer uma previsão centrada na média. A classificação média do utilizador alvo é então adicionada de volta a esta previsão para fornecer uma previsão de classificação bruta $\hat{r}_{ui}$ do utilizador alvo *u*, para o item *i* [3], ou seja,

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v).(r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u,v)} \tag{3.2}$$

onde μ_u é a classificação média para cada utilizador *u* e μ_v é a classificação média para cada utilizador *v*.

Também existe uma versão baseada em item desse algoritmo, na qual verifica-se os *k* itens mais próximos avaliados pelo utilizador *u* e usamos suas classificações e semelhanças com o item *i* para prever a classificação do item *i* [28].

Algoritmos baseados na factorização de matriz (*Matrix Fatorization*)

Em sua forma básica, a factorização da matriz caracteriza itens e utilizadores por vetores de fatores inferidos dos padrões de classificação de itens. A alta correspondência entre os fatores do item e do utilizador leva a uma recomendação. Esses algoritmos se tornaram populares nos últimos anos, combinando boa escalabilidade com precisão preditiva. Além disso, oferecem muita flexibilidade para modelar várias situações da vida real. Os sistemas de recomendação contam com diferentes tipos de dados de entrada, que geralmente são colocados em uma matriz com uma dimensão representando os utilizadores e a outra dimensão representando os itens de interesse. Os dados mais convenientes são as classificações explícitas de alta qualidade, que inclui a entrada explícita de utilizadores sobre seu interesse nos produtos [29].

Singular Value Decomposition (SVD): Os modelos de factorização de matriz mapeiam utilizadores e itens para um espaço de fator latente comum de dimensionalidade *f*, de modo que as interações utilizador-item são modeladas como produtos internos nesse espaço. O espaço latente tenta explicar as classificações, caracterizando produtos e utilizadores em fatores inferidos automaticamente do feedback do utilizador. Assim, cada item i está associado a um vetor $q_i \in \mathbb{R}^f$, e cada utilizador *u* está associado a um vetor $p_u \in \mathbb{R}^f$. Para um determinado item *i*, os elementos de q_i medem até que ponto o item possui esses fatores, positivos ou negativos. Para um determinado utilizador *u*, os elementos de p_u medem a extensão do interesse que o utilizador tem em itens que são altos nos fatores correspondentes, novamente, eles podem ser positivos ou negativos. O produto escalar resultante, $q_i^T p_u$, captura a interação entre o utilizador *u* e o item *i*, ou seja, o interesse geral do utilizador nas características do item. A avaliação final é criada adicionando também os preditores de linha de base mencionados que dependem apenas do utilizador ou item. Assim, uma classificação é prevista pela equação [29], [1]

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \quad (3.3)$$

onde: μ – média global; b_u – tendência do utilizador; b_i – tendência do item e $q_i^T p_u$ – interação do utilizador com o item.

A fim de aprender os parâmetros do modelo (b_u, b_i, p_u e q_i), o erro quadrático regularizado é minimizado:

$$min_{p*,q*,b*} \sum_{(u,i)\in K} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left({b_i}^2 + {b_u}^2 + \left|\left|q_i\right|\right|^2 + \left|\left|p_u\right|\right|^2 \right) \quad (3.4)$$

Uma classificação r_{ui} indica a preferência do utilizador *u* do item *i*, onde valores altos significam preferência mais forte. Distinguimos as classificações previstas das conhecidas, usando a notação $\hat{r}_{ui}$ para o valor previsto de r_{ui}. Os pares (*u, i*) pelos quais r_{ui} é conhecido são armazenados no conjunto K e a constante λ controla a extensão da regularização, sendo geralmente determinada por validação cruzada. A minimização é normalmente realizada por gradiente descendente estocástico (*stochastic gradient descent*) ou mínimos quadrados alternados (*alternating least squares*) [29].

Singular Value Decomposition Plus Plus (SVD++): é uma extensão do SVD e tem em consideração as classificações implícitas. A precisão da previsão é aprimorada considerando também o *feedback* implícito, que fornece uma indicação adicional das preferências do utilizador. Isso é especialmente útil para os utilizadores que forneceram muito mais *feedback* implícito do que explícito. Mesmo nos casos em que o *feedback* implícito independente está ausente, pode-se capturar um sinal significativo ao considerar quais itens os utilizadores avaliam, independentemente de seu valor de classificação. Isso levou a vários métodos que modelaram um fator do utilizador pela identidade dos itens que ele classificou. Um deles é o SVD++, que mostrou oferecer acurácia superior ao SVD. Para este fim, um segundo conjunto de fatores de itens é adicionado, relacionando cada item *i*, a um vetor de fatores $y_i \in \mathbb{R}^f$. Esses novos fatores de item são usados para caracterizar os utilizadores com base no conjunto de itens que eles classificaram. O modelo exato é o seguinte [1]:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |R_u|^{-\frac{1}{2}} \sum_{j \in R_u} y_j \right) \tag{3.5}$$

O conjunto R_u contém os itens avaliados pelo utilizador *u*. Agora, um utilizador *u* é modelado como $p_u + |R_u|^{-\frac{1}{2}} \sum_{j \in R_u} y_j$. Utiliza-se um vetor livre de fatores do utilizador, p_u, que é aprendido a partir da classificação explícita fornecida. Esse vetor é complementado pela adição de $|R_u|^{-\frac{1}{2}} \sum_{j \in R_u} y_j$, que representa a perspectiva de *feedback* implícito.

Non-Negative Matrix Fatorization (NMF): é um algoritmo de matriz de factorização, no qual a matriz de utilizador-item é decomposta em fatores de utilizador e item. A condição, entretanto, é que a matriz do utilizador-item e os fatores devem todos ter valores não negativos. Os fatores de utilizador e item são inicializados com valores aleatórios e a otimização é feita por gradiente descendente estocástico (*stochastic gradient descent - SGD*). Este algoritmo é altamente dependente dos valores inicializados para os fatores. As *baselines* do utilizador e do item também podem ser incorporadas, mas o modelo torna-se suscetível a sobredimensionamento, que, entretanto, pode ser controlado por uma boa escolha do parâmetro de regularização [30]. A previsão $\hat{r}_{ui}$ é definida como:

$$\hat{r}_{ui} = q_i^T p_u \tag{3.6}$$

Em cada etapa do procedimento SGD, os fatores f ou utilizador u e item i são atualizados da seguinte forma:

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r}_{ui} + \lambda_u |I_u| p_{uf}} \tag{3.7}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r}_{ui} + \lambda_i |U_i| q_{if}} \tag{3.8}$$

onde λu e λi são as constantes de regularização.

SlopeOne

Funciona com base no princípio intuitivo de um diferencial de popularidade entre itens para os utilizadores. Em pares, ele determina o quão melhor um item é apreciado do que outro. Uma forma de medir esse diferencial é simplesmente subtrair a classificação média dos dois itens [2]. Por sua vez, essa diferença pode ser usada para prever a classificação de outro utilizador de um desses itens, dada a classificação do outro. Muitos desses diferenciais existem em um conjunto de treinamento para cada classificação desconhecida e tomamos uma média desses diferenciais. Assim, dado um conjunto de treinamento, e quaisquer dois itens *i* e *j* com classificações r_{ui} e r_{uj} respetivamente em alguma avaliação do utilizador *u* consideramos o desvio médio do item *i* em relação ao item *j* como [31]:

$$dev(i,j) = \frac{1}{|U_{ij}|} + \sum_{u \in U_{ij}} r_{ui} - r_{uj} \tag{3.9}$$

O algoritmo considera que qualquer avaliação do utilizador *u* que não contenha as classificações r_{ui} e r_{uj} não será incluída no somatório. U_{ij} é o conjunto de todos os utilizadores que classificaram os itens *i* e *j*. A previsão de uma classificação $\hat{r}_{ui}$ é definida como [31]:

$$\hat{r}_{ui} = \frac{1}{|R_{i(u)}|} + \sum_{j \in R_{i(u)}} dev(i,j) \tag{3.10}$$

Co-Clustering

Clustering é uma técnica de aprendizagem não supervisionada. Normalmente, um algoritmo de *clustering* busca agrupar objetos semelhantes. A abordagem de *Co-Clustering* é semelhante às técnicas de correlação e baseadas em *clustering* no sentido de que as vizinhanças são empregues para previsão, a principal diferença é que os utilizadores e os itens são agrupados de forma que a sinonímia de itens deixe de ser um problema. A ideia principal desse algoritmo é obter, simultaneamente, vizinhanças de utilizadores e itens por meio de *co-clustering* e gerar previsões com base nas classificações médias dos *co-clusters* (vizinhanças de itens de utilizadores), levando em consideração as tendências individuais dos utilizadores e itens. Basicamente, utilizadores

e itens são atribuídos a alguns *clusters* C_u, C_i, e alguns co-clusters C_{ui}. A previsão $\hat{r}_{ui}$ é definido como [32]:

$$\hat{r}_{ui} = \overline{C_{ui}} + (\mu_u - \overline{C_u}) + (\mu_i - \overline{C_i}) \quad (3.11)$$

onde $\overline{C_{ui}}$ é a classificação média do *co-cluster* C_{ui}, $\overline{C_u}$ é a classificação média do cluster C_u, e é a classificação média do C_i. Se o utilizador for desconhecido, a previsão é $\hat{r}_{ui} = \mu_i$. Se o item for desconhecido, a previsão é $\hat{r}_{ui} = \mu_u$. Se o uutilizador e o item forem desconhecidos, a previsão é $\hat{r}_{ui} = \mu$ [32].

3.4. Métricas de Avaliação

Ao implementar um sistema de recomendação, é útil ser capaz de avaliar o desempenho do sistema. Na literatura de sistemas de recomendação, é comum usar avaliação offline para algoritmos de pré-teste para entender seu comportamento antes do teste do utilizador. A avaliação offline também é benéfica para realizar comparações diretas e objetivas de diferentes algoritmos de forma reproduzível.

A pesquisa de sistemas de recomendação tem utilizado vários tipos de métricas para avaliar a qualidade de um sistema de recomendação. Utilizamos métricas que calculam o erro entre a classificação real e a classificação prevista do utilizador, tais como MAE e RMSE. A partir dessas métricas é possível inferir quão perto estão os valores previstos das classificações reais dadas pelos utilizadores. No entanto, as métricas de precisão não são adequadas para medir o desempenho de classificação dos sistemas de recomendação. Portanto, o estudo também conduziu uma avaliação com base nas métricas de classificação *Precision@k, Recall@k e F1@k*. Essas métricas têm em conta o número de *true positives* (TP), *false negative* (FN)*, false positives* (FP) *e true negatives* (TN) presentes na lista de recomendação.

MAE: O Erro Médio Absoluto ou *Mean Absolute Error*, é a média do desvio das recomendações de seus valores reais especificados pelo utilizador. É a média sobre a amostra de teste das diferenças absolutas entre a previsão e a observação real, onde todas

as diferenças individuais têm peso igual. Quanto menor o MAE, mais precisamente o mecanismo de recomendação prevê as classificações do utilizador [33]:

$$MAE = \frac{\sum_{u,i} |\hat{r}_{ui} - r_{u,i}|}{n} \tag{3.12}$$

onde *n* é o conjunto de teste, $r_{u,i}$ representa as classificações reais do utilizador.

RMSE: A Raiz Quadrada do Erro Quadrático Médio ou *Root Mean Squared Error*, é a raiz quadrada da média da diferença entre as classificações previstas e as classificações reais, ou seja, o sistema gera classificações previstas $\hat{r}_{ui}$ para um conjunto de teste *n* de pares de itens de utilizadores (u, i) para os quais as classificações reais $r_{u,i}$ são conhecidas [34]. O RMSE entre as classificações previstas e reais é dado por:

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}\left(\hat{r}_{ui} - r_{u,i}\right)^2} \tag{3.13}$$

Precision@k: é a proporção de itens recomendados no conjunto top-k que são relevantes. É definido como [3]:

$$Precison@k = \frac{|\mathrm{R} \cap \widehat{R_{1:k}}|}{k} = \tag{3.14}$$

Recall@k: é a fração de itens relevantes recomendados. É calculado conforme a equação 3.15 [3]:

$$Recall@k = \frac{|\mathrm{R} \cap \widehat{R_{1:k}}|}{|R|} \tag{3.15}$$

onde $\widehat{R_{1:k}}$ denota os primeiros *k* itens previstos para um utilizador.

F1@k: é uma forma de combinar *precision@k* e *recall@k* do modelo, e é definido como a média harmônica da *precision* e *recall* do modelo. É calculado com base na equação 3.16 [3]:

$$F1@k = \frac{2 \text{ x precision@k x recall@k}}{\text{precision@k} + \text{recall@k}} \tag{3.16}$$

onde *k* refere-se aos valores em uma lista de recomendações top-*k*.

Capítulo 4 – Análise e Discussão dos Resultados

Neste capítulo, analisamos os resultados do desempenho dos algoritmos descritos no capítulo 3. Os algoritmos propostos foram avaliados em termos de MAE, RMSE, *Precision@k*, *Recall@k* e *F1@k*, permitindo dessa forma uma comparação em termos de desempenho.

Antes da implementação dos algoritmos e seguindo a abordagem descrita na seção 3.3, dividimos o conjunto de dados em treinamento e teste. O conjunto de dados tem um total de 2.794.726 linhas, cada uma com informações exclusivas sobre as interações do utilizador-item.

Dividiu-se, esse conjunto na proporção de 80/20. Sendo 80% para o conjunto de treinamento (2.235.780 linhas) e 20% para o conjunto de teste (558.946 linhas). Esse último conjunto foi subdivido em conjunto de consulta 10% e conjunto de destino 10%. O conjunto de treinamento juntamente com o conjunto de consulta foram utilizados para implementar os algoritmos de recomendação de filtragem colaborativa. Uma vez implementados, avaliamos os algoritmos com o conjunto de dados de teste.

É possível observar na Figura 5, que as classificações estão distribuídas quase uniformemente, em ambos os conjuntos, exceto a classificação de valor 1 que está ligeiramente acima das outras classificações.

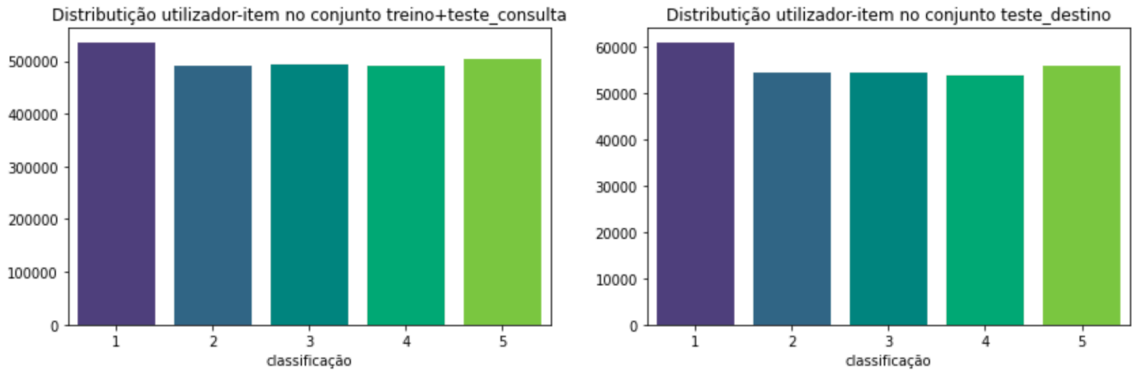

Figura 5 - Divisão do conjunto de dados em treinamento e teste

4.1. Resultados das Métricas de Avaliação – RMSE e MAE

Calculámos o RMSE e o MAE para avaliar a desempenho, para todos os algoritmos de filtragem colaborativa que nos propusemos a implementar, usando a validação cruzada *k-fold* com $k = 5$.

Nas Figuras 6-12 podemos verificar os resultados da validação cruzada de cada algoritmo.

```
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2277  1.2259  1.2267  1.2275  1.2270  1.2270  0.0006
MAE (testset)     1.0169  1.0149  1.0154  1.0165  1.0159  1.0159  0.0007
Fit time          659.54  702.43  712.18  712.35  720.55  701.41  21.71
Test time         1513.76 1573.00 1544.62 1560.75 1550.60 1548.55 19.88
```

Figura 6 - Validação cruzada k-fold (k=5) KNNBasic

```
Evaluating RMSE, MAE of algorithm KNNWithMeans on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2472  1.2475  1.2497  1.2499  1.2473  1.2483  0.0012
MAE (testset)     1.0352  1.0340  1.0367  1.0371  1.0347  1.0356  0.0012
Fit time          702.86  749.12  789.22  790.19  792.51  764.78  34.91
Test time         1727.09 1754.07 1728.63 1760.81 1741.73 1742.47 13.41
```

Figura 7 - Validação cruzada k-fold (k=5) KNNWithMeans

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2403  1.2382  1.2389  1.2381  1.2376  1.2386  0.0009
MAE (testset)     1.0066  1.0047  1.0049  1.0047  1.0036  1.0049  0.0009
Fit time          155.76  157.31  155.36  149.99  148.50  153.38  3.48
Test time         11.93   11.14   11.48   11.79   10.92   11.45   0.38
```

Figura 8 - Validação cruzada k-fold (k=5) SVD

```
Evaluating RMSE, MAE of algorithm SVDpp on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2180  1.2186  1.2189  1.2166  1.2174  1.2179  0.0008
MAE (testset)     0.9723  0.9726  0.9747  0.9723  0.9718  0.9727  0.0010
Fit time          4951.41 4890.45 4744.60 4923.61 5008.75 4903.77 88.54
Test time         110.06  110.36  110.73  110.77  109.80  110.35  0.38
```

Figura 9 - Validação cruzada k-fold (k=5) SVD++

```
Evaluating RMSE, MAE of algorithm NMF on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2232  1.2228  1.2246  1.2226  1.2212  1.2229  0.0011
MAE (testset)     1.0012  1.0013  1.0025  1.0004  0.9999  1.0011  0.0009
Fit time          155.38  155.83  154.81  145.40  142.71  150.83  5.60
Test time         11.08   10.07   11.00   9.88    9.14    10.24   0.73
```

Figura 10 - Validação cruzada k-fold (k=5) NMF

```
Evaluating RMSE, MAE of algorithm SlopeOne on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.1986  1.1969  1.1973  1.1993  1.1963  1.1977  0.0011
MAE (testset)     0.9843  0.9825  0.9823  0.9844  0.9820  0.9831  0.0010
Fit time          24.20   24.74   25.25   25.03   23.95   24.63   0.49
Test time         78.07   77.13   79.43   75.91   72.14   76.54   2.48
```

Figura 11 - Validação cruzada k-fold (k=5) SlopeOne

```
Evaluating RMSE, MAE of algorithm CoClustering on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.2374  1.2376  1.2323  1.2345  1.2352  1.2354  0.0020
MAE (testset)     1.0233  1.0233  1.0184  1.0201  1.0211  1.0212  0.0019
Fit time          57.75   59.34   59.28   59.16   60.18   59.14   0.78
Test time         9.38    8.13    9.38    8.97    8.15    8.80    0.56
```

Figura 12 - Validação cruzada k-fold (k=5) Co-Clustering

Apresentamos, nas Figuras 13 e 14, os resultados das métricas RMSE e MAE dos diferentes algoritmos de recomendação para comparação. Verificamos que para esse conjunto de dados os algoritmos não apresentam valores muito diferentes entre si no cálculo dessas métricas. Contudo, os algoritmos que exibiram os melhores resultados foram o SlopeOne e o SVD++. Na comparação do RMSE, o SlopeOne exibiu o melhor resultado quando comparado com os outros algoritmos, seguido do SVD++. Com as classificações previstas a desviarem-se das classificações reais ≈ 1,1977 para o SlopeOne. E para o SVD++ ≈ 1,2179. Na observação do MAE, SVD++ apresentou o menor valor de MAE ≈ 0,9727 e logo em seguida o SlopeOne ≈ 0,9831.

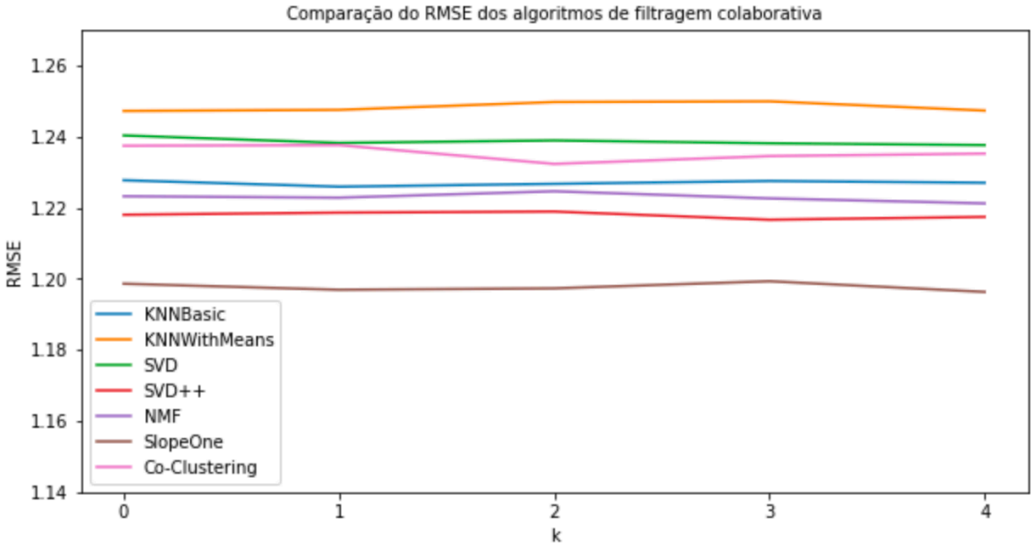

Figura 13 - Comparação do RMSE dos algoritmos de recomendação

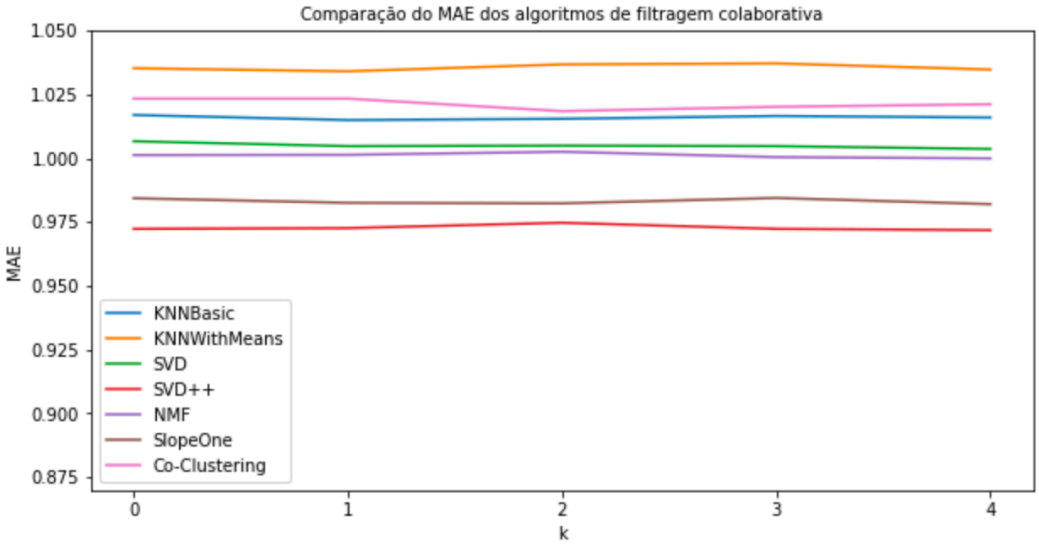

Figura 14 - Comparação do MAE dos algoritmos de recomendação

Na análise das métricas RMSE e MAE, o KKNWithMeans teve a pior avaliação, obtendo o valor mais alto em ambas as métricas. Ficando o SlopeOne com a melhor avaliação do RMSE e o SVD++ com a melhor avaliação do MAE.

Verificámos, também, a distribuição das classificações previstas, nas Figuras 16-22, resultantes da implementação dos algoritmos, em comparação com as classificações reais do conjunto de teste, na Figura 15. A distribuição das classificações previstas no conjunto de teste é visivelmente diferente para todos os algoritmos. Mostra que o sistema de

recomendação não é perfeito e não reflete tão bem quanto o desejado, a distribuição real das classificações implícitas dos videojogos.

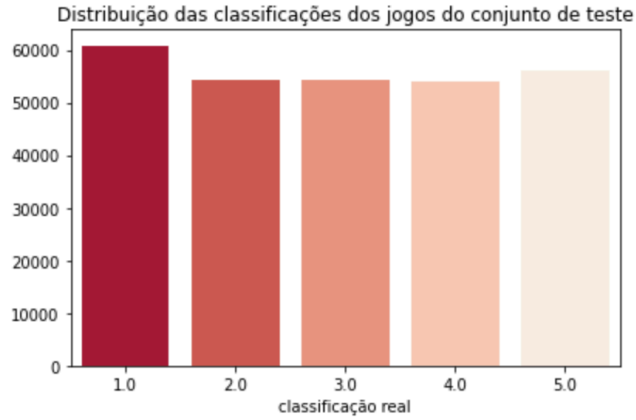

Figura 15 - Distribuição das classificações reais

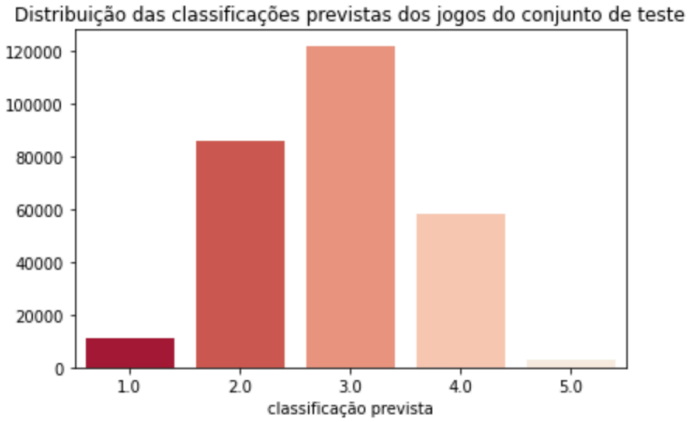

Figura 16 - Distribuição das classificações previstas utilizando KKNBasic

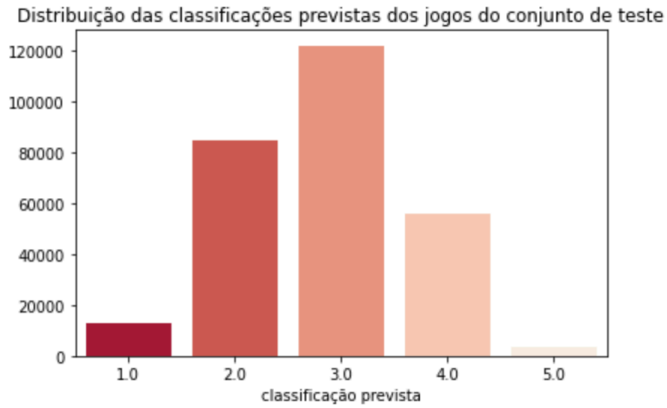

Figura 17 - Distribuição das classificações previstas utilizando KKNWithMeans

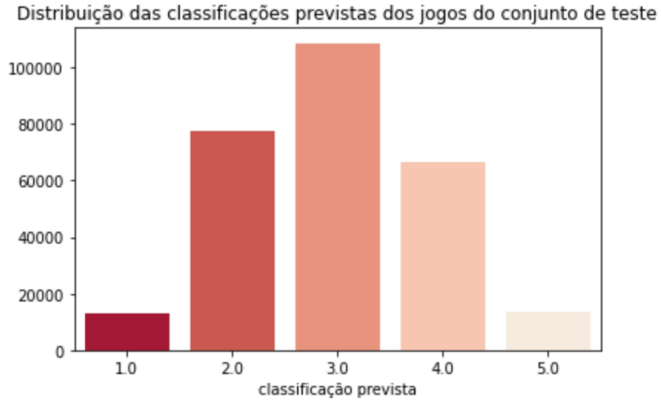

Figura 18 - Distribuição das classificações reais e previstas utilizando SVD

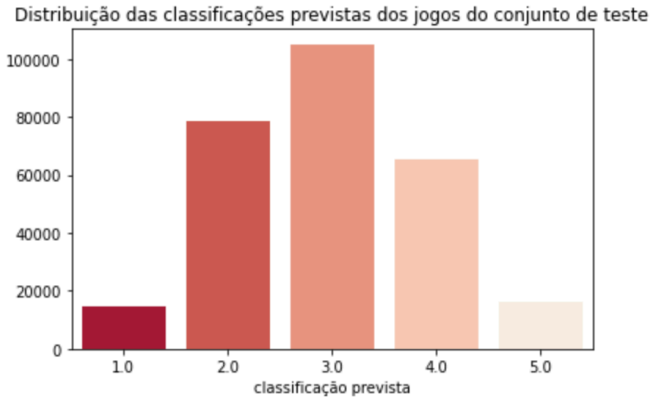

Figura 19 - Distribuição das classificações reais e previstas utilizando SVD++

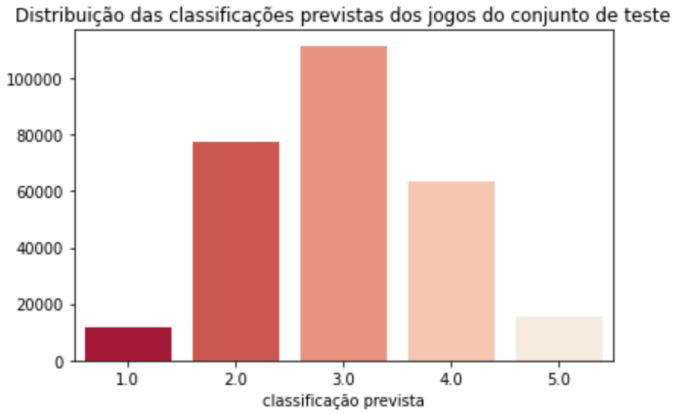

Figura 20 - Distribuição das classificações reais e previstas utilizando NMF

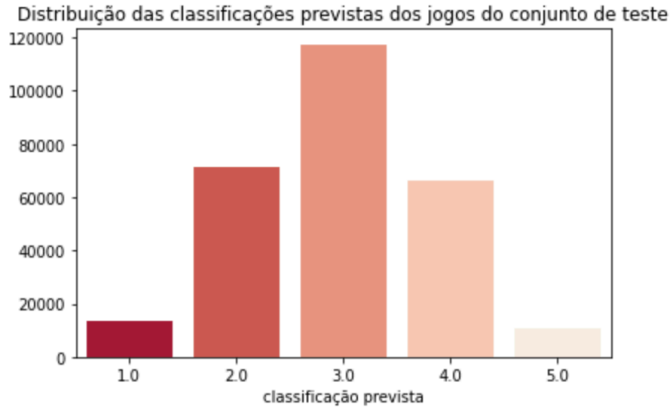

Figura 21 - Distribuição das classificações reais e previstas utilizando SlopeOne

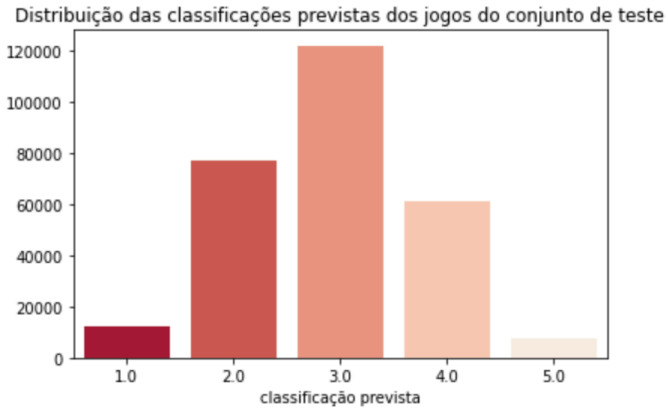

Figura 22 - Distribuição das classificações reais e previstas utilizandoCo-Clustering

4.2. Resultados das Métricas de Avaliação – *Precision*, *Recall* e *F-measure*

A *precision@k* mostra para cada algoritmo o quão boas são as recomendações *top-k*. Uma alta pontuação de *precision@k* indica que a maioria de nossas recomendações são relevantes, o que está de acordo com nossos objetivos. Também consideramos o *recall@k* para inquirir a proporção de recomendações relevantes que conseguimos fazer.

O denominador de *recall@k* é independente de *k*, portanto, esperamos que *recall@k* aumente à medida que *k* aumenta. Do mesmo modo que, esperamos que *precision@k* diminua à medida que *k* aumenta. Observamos essas tendências em todos os algoritmos treinados e apresentados nas Figuras 23-29:

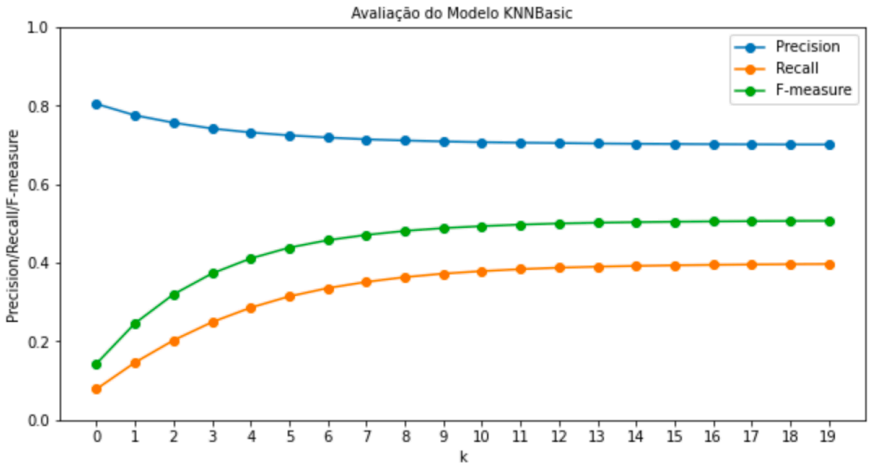

Figura 23 - Medições do Precision, Recall, F-measure do KNNBasic

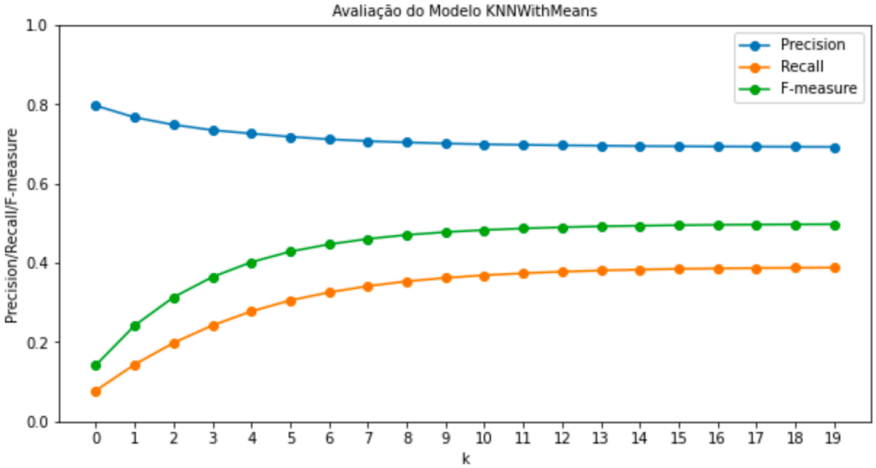

Figura 24 - Medições do Precision, Recall, F-measure do KNNWithMeans

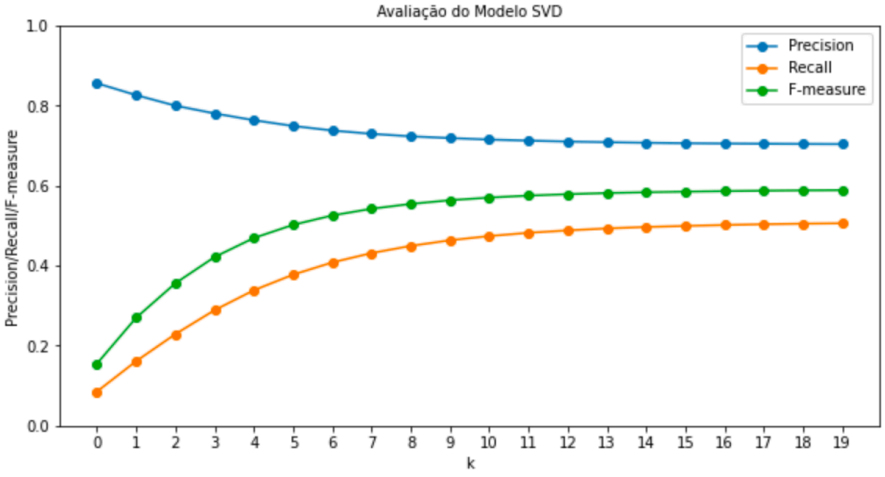

Figura 25 - Medições do Precision, Recall, F-measure do SVD

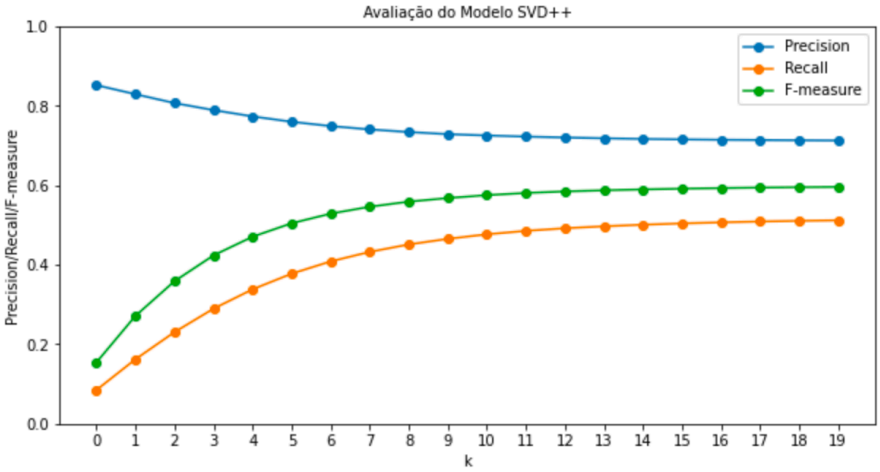

Figura 26 - Medições do Precision, Recall, F-measure do SVD++

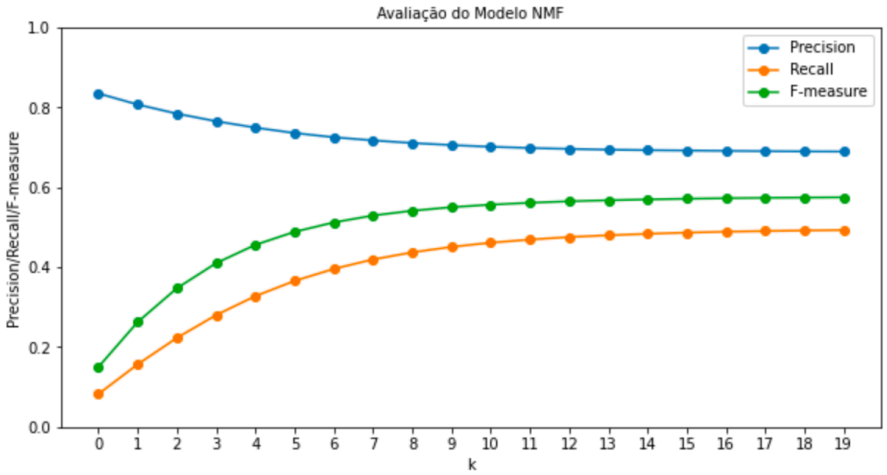

Figura 27 - Medições do Precision, Recall, F-measure do NMF

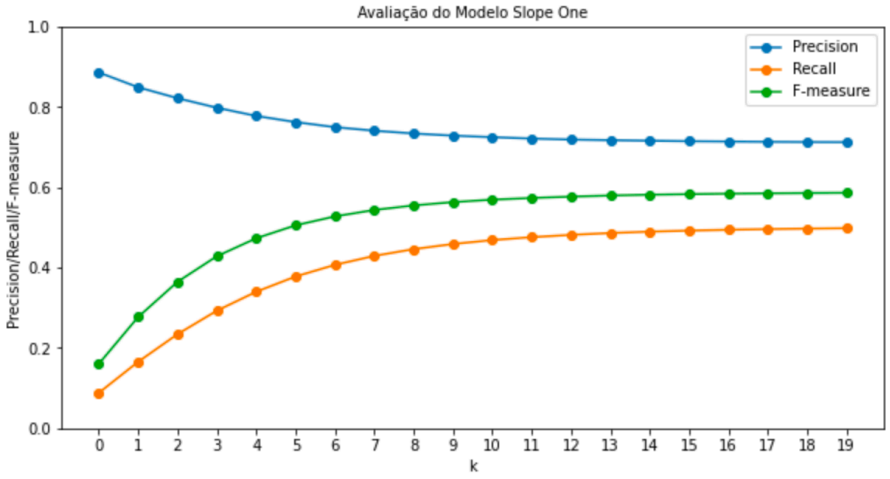

Figura 28 - Medições do Precision, Recall, F-measure do SlopeOne

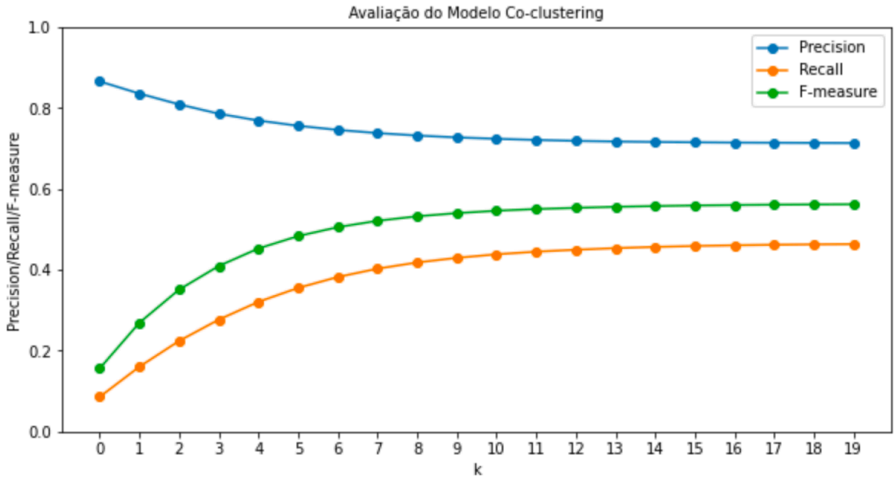

Figura 29 - Medições do Precision, Recall, F-measure do Co-Clustering

Em sistemas de recomendação é bastante comum recomendarmos uma lista de itens, top-*k*, ao utilizador. Por isso, calculamos p*recison@k* e *recall@k* nos primeiros *k* jogos recomendados, para medir o desempenho dos algoritmos de filtragem colaborativa. Analisamos o valor de *k* entre 1 e 20.

Tabela 3 - Valores do precision@k, com k = 5, 10, 20

	precision@5	*precision@10*	*precision@20*
KNNBasic	0.7374	0.7144	0.7064
KNNWithMeans	0.7293	0.7043	0.6955
SVD	0.7619	0.7162	0.7003
SVD++	0.7757	0.7309	0.7145
NMF	0.7525	0.7073	0.6920
SlopeOne	0.7840	0.7336	0.7183
Co-clustering	0.7713	0.7288	0.7150

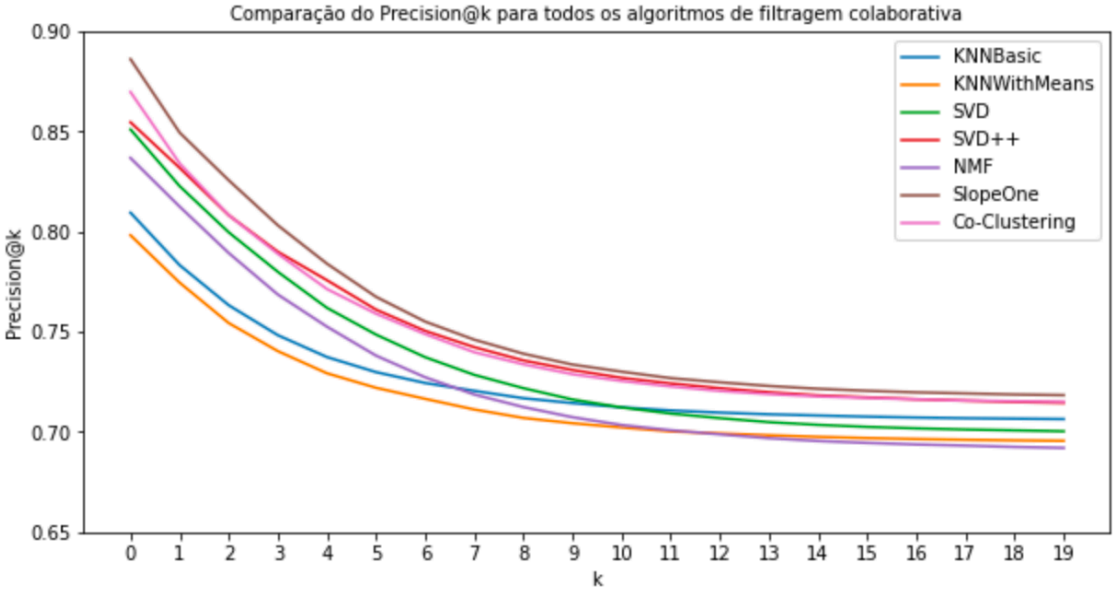

Figura 30 - Comparação do precision@k para todos os algoritmos de recomendação

Na Figura 30 e na Tabela 3, comparamos a *precision@k* para diferentes *k* no conjunto de teste destino. No geral, o SlopeOne obteve o melhor resultado. Quando *precision@5* significa que em 5 jogos recomendados, 78% das recomendações feitas com o SlopeOne são relevantes para o utilizador. E para *precision@20*, temos que em 20 jogos recomendados, são relevantes 71% das recomendações.

Tabela 4 - Valores do recall@k, com k = 5, 10, 20

	recall@5	*recall@10*	*recall@20*
KNNBasic	0.2866	0.3718	0.3954
KNNWithMeans	0.2767	0.3595	0.3841
SVD	0.3367	0.4609	0.5028
SVD++	0.3397	0.4640	0.5095
NMF	0.3275	0.4470	0.4882
SlopeOne	0.3391	0.4535	0.4920
Co-clustering	0.3174	0.4229	0.4558

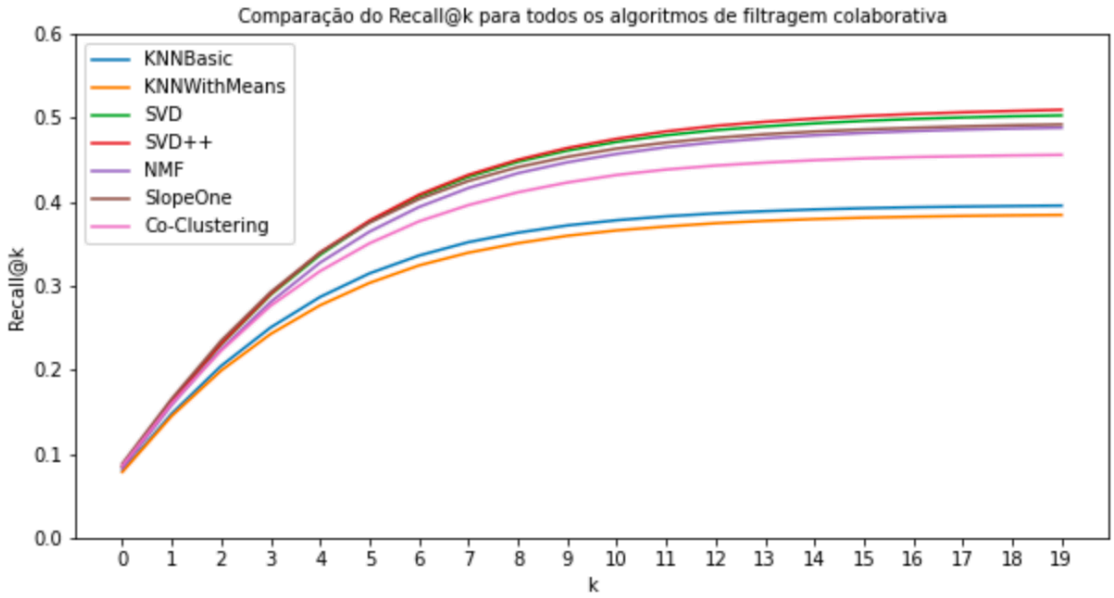

Figura 31 - Comparação do recall@k para todos os algoritmos de recomendação

A Figura 31 e a Tabela 4, mostram a comparação do *recall@k* de todos os algoritmos de recomendação. Ao compararmos essa métrica verificámos que o SVD++ teve melhor desempenho, embora não haja muita diferença com o SVD. No SVD++, para um *recall@20*, temos que aproximadamente 50% do número total de jogos relevantes aparecem nos primeiros *k* resultados da lista de recomendações.

Tabela 5 - Valores do F1@k, com k = 5, 10, 20

	F1@5	*F1@10*	*F1@20*
KNNBasic	0.4128	0.4891	0.5070
KNNWithMeans	0.4012	0.4760	0.4951
SVD	0.4670	0.5608	0.5854
SVD++	0.4725	0.5677	0.5948
NMF	0.4564	0.5478	0.5725
SlopeOne	0.4735	0.5605	0.5840
Co-clustering	0.4498	0.5352	0.5567

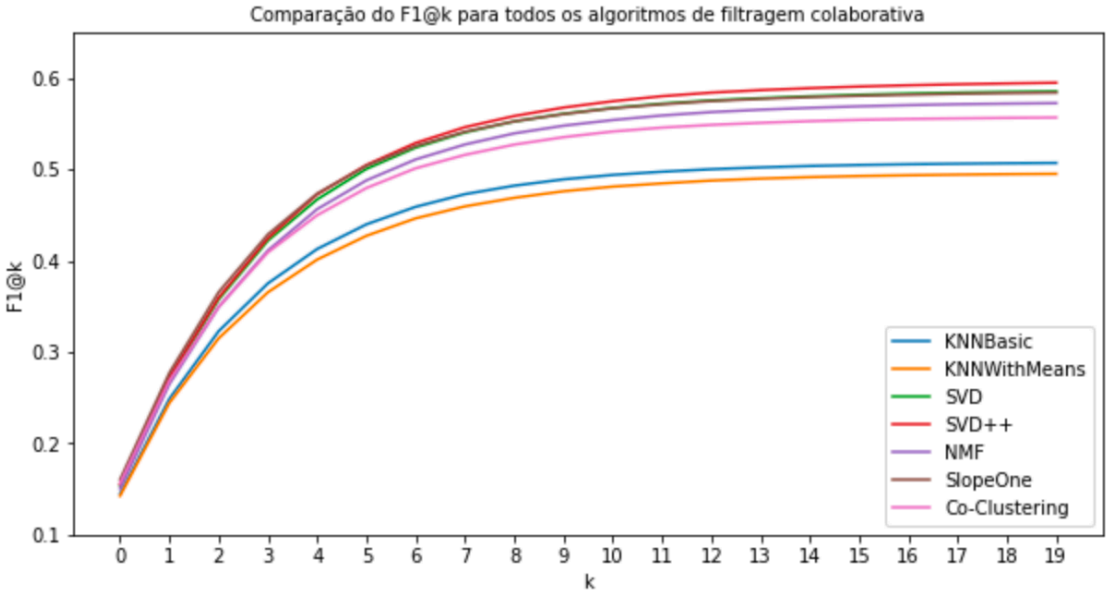

Figura 32 - Comparação do F1@k para todos os algoritmos de recomendação

De forma geral, nas Figuras 30, 31 e 32, observamos que os algoritmos SlopeOne e o SVD++ tiveram os melhores resultados, tal como ocorreu na avaliação com o RMSE e o MAE. Esses algoritmos têm um desempenho melhor do que os outros quando comparados em todas as métricas de avaliação. Podemos referir que o valor de *k* tem grande impacto na precisão da previsão, ou seja, na qualidade da recomendação. Portanto, é necessário ponderarmos sobre o valor desse parâmetro para decidirmos por um algoritmo em detrimento de outro.

	Tempo médio de treino (segundos)	*Tempo médio de teste (segundos)*
KNNBasic	701,41	1548,55
KNNWithMeans	764,78	1742,47
SVD	153,38	11,45
SVD++	4903,77	110,35
NMF	150,83	10,24
SlopeOne	24,63	76,54
Co-clustering	59,14	8,80

Tabela 6 - Tempo médio, em segundos, de treino e teste

Na Tabela 6 temos o tempo médio de execução de treino e teste dos algoritmos de recomendação avaliados, em segundos. O SlopeOne possui o tempo de treino mais baixo, ao contrário do SVD++ que tem o tempo de treino mais elevado. Ambos os algoritmos têm os melhores desempenhos quando analisamos as métricas de avaliação que consideramos neste trabalho. Ao examinarmos o tempo de execução de ambos, escolhemos o SlopeOne, já que apresenta exigência computacional inferior ao SVD++.

Capítulo 5 – Conclusões e Recomendações

Este trabalho trata do tema dos sistemas de recomendação no domínio dos videojogos. A abordagem proposta consiste numa comparação de algoritmos de recomendação baseados em filtragem colaborativa, que utiliza dados dos utilizadores da plataforma Steam. Efetuamos tarefas de pré-processamento dos dados antes do treinamento dos algoritmos de recomendação, sendo feita também a conversão da classificação implícita em classificação explícita. Essa conversão permitiu-nos utilizar o tempo de jogo total como classificação implícita, transformando-o numa classificação numérica de 1 a 5.

Implementámos abordagens baseadas em sete algoritmos de recomendação, com recurso à biblioteca *Surprise*, com a validação cruzada *k-fold*=5, treinamos cada algoritmo e validamos por cinco vezes. As métricas de avaliação como RMSE, MAE, *precision@k, recall@k e F1@k,* ajudaram-nos a avaliar o desempenho desses algoritmos de recomendação e verificámos qual deles teve o melhor desempenho em comparação com os outros.

Analisando os resultados dessas métricas o SlopeOne e o SVD++ mostraram ser os melhores algoritmos de recomendação. Se tivermos em conta o custo computacional de cada algoritmo de recomendação, então optamos pelo SlopeOne, no geral. Entretanto, entendemos que é preciso, também, ter em conta o tamanho da lista de itens a recomendar, numa recomendação de top-*k* tem impacto direto nos cálculos do *precision@k* e do *recall@k.*

5.1. Respostas às Questões de Investigação

Qual o desempenho dos algoritmos de recomendação em comparação entre si, quando aplicámos dados implícitos aos mesmos?

Ao analisarmos os resultados das métricas percebemos que não há para o nosso conjunto de dados uma diferença tão significativa entre os algoritmos, embora os resultados indiquem que o SlopeOne e o SVD++ são os algoritmos que apresentam melhor desempenho. Depois de considerarmos também os recursos e tempo do sistema necessário para as suas execuções, observamos que o SlopeOne é uma boa escolha de um modo geral.

Considerando que o tempo de jogo de um utilizador é um dado implícito do sistema, o mesmo pode servir como uma boa representação das preferências do utilizador?

Evidenciamos que é realmente possível usar o tempo de jogo para inferir classificações explícitas para fazer recomendações no domínio de videojogos e especificamente no Steam plataforma. No entanto temos em conta que é necessário ter em atenção como transformar o tempo de jogo em classificação, pois a boa representação das preferências do utilizador está diretamente relacionada ao desempenho do sistema de recomendação.

5.2. Propostas de Investigação Futura

Os algoritmos apresentados, além de serem de fácil aplicabilidade e menor custo computacional quando comparados com algoritmos mais complexos, podem produzir boas recomendações. Para futuros trabalhos pode ser útil analisar outros métodos de transformação do tempo de jogo em classificação explícita e, talvez, tentar extrair as *reviews* dos utilizadores a fim de inferir melhores classificações explícitas. Pode ser, também, interessante testar outras métricas de avaliação como: *Coverage* que nos poderá informar a proporção de jogos recomendados em relação ao total de jogos, o que é atraente perceber se o sistema de recomendação, para além de ter um bom desempenho, tem também alta cobertura; *Synonymy* para verificar a existência de jogos iguais ou semelhantes que possam ter nomes diferentes, mas é preciso ter em atenção que versões diferentes de um jogo equivalem a jogos diferentes, como por exemplo: *BioShock* e *BioShock 2.*

Bibliografia

[1] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. Springer Publishing Company, Incorporated, 2011.

[2] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommendation system: An Introduction*, vol. 91. New York: Cambridge University Press, 2011.

[3] C. C. Aggarwal, *Recommender Systems: The Textbook*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[4] K. Najmani, E. habib Benlahmar, N. Sael, and A. Zellou, "A comparative study on recommender systems approaches," *BDIoT'19 Proc. 4th Int. Conf. Big Data Internet Things*, vol. 3, pp. 1–5, 2019, doi: 10.1145/3372938.

[5] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6. pp. 734–749, 2005, doi: 10.1109/TKDE.2005.99.

[6] J. Wei, J. He, Z. Tang, K. Chen, and Y. Zhou, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Syst. Appl.*, vol. 69, 2017, doi: 10.1016/j.eswa.2016.09.040.

[7] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative filtering recommender systems," *Found. Trends Human-Computer Interact.*, vol. 4, no. 2, pp. 81–173, 2010, doi: 10.1561/1100000009.

[8] K. Shah, A. Salunke, S. Dongare, and K. Antala, "Recommender systems: An overview of different approaches to recommendations," *Proceedings of 2017 International Conference on Innovations in Information, Embedded and Communication Systems, ICIIECS 2017*, vol. 2018-Janua. pp. 1–4, 2018, doi: 10.1109/ICIIECS.2017.8276172.

[9] D. Cao *et al.*, "Version-sensitive mobile App recommendation," *Inf. Sci. (Ny).*, vol. 381, pp. 161–175, 2017, doi: 10.1016/j.ins.2016.11.025.

[10] P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds., *The Adaptive Web: Methods and Strategies of Web Personalization*. Berlin, Heidelberg: Springer-Verlag, 2007.

[11] J. Pérez-Marcos, L. Martín-Gómez, D. M. Jiménez-Bravo, V. F. López, and M. N. Moreno-García, "Hybrid system for video game recommendation based on implicit ratings and social networks," *J. Ambient Intell. Humaniz. Comput.*, 2020, doi: 10.1007/s12652-020-01681-0.

[12] D. Jannach, L. Lerche, and M. Zanker, "Recommending based on implicit feedback," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10100 LNCS, pp. 510–569, 2018, doi: 10.1007/978-3-319-90092-6_14.

[13] G. Jawaheer, M. Szomszor, and P. Kostkova, "Comparison of implicit and explicit feedback from an online music recommendation service," *Proc. 1st Int. Work. Inf. Heterog. Fusion Recomm. Syst. HetRec 2010, Held 4th ACM Conf. Recomm. Syst. RecSys 2010*, pp. 47–51, 2010, doi: 10.1145/1869446.1869453.

[14] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3. Ministry of Higher Education and Scientific Research, pp. 261–273, 2015, doi: 10.1016/j.eij.2015.06.005.

[15] R. Cañamares, P. Castells, and A. Moffat, "Offline evaluation options for recommender systems," *Inf. Retr. J.*, vol. 23, no. 4, pp. 387–410, 2020, doi: 10.1007/s10791-020-09371-3.

[16] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *J. Mach. Learn. Res.*, vol. 10, pp. 2935–2962, 2009.

[17] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," *arXiv*, 2019.

[18] D. Wang, M. Moh, and T. S. Moh, "Using deep learning and steam user data for better video game recommendations," *ACMSE 2020 - Proc. 2020 ACM Southeast Conf.*, pp. 154–159, 2020, doi: 10.1145/3374135.3385283.

[19] G. Cheuque, J. Guzmán, and D. Parra, "Recommender systems for online video game platforms: The case of steam," *Web Conf. 2019 - Companion World Wide Web Conf. WWW 2019*, vol. 2, pp. 763–771, 2019, doi: 10.1145/3308560.3316457.

[20] S. M. Anwar, T. Shahzad, Z. Sattar, R. Khan, and M. Majid, "A game

recommender system using collaborative filtering (GAMBIT)," *Proc. 2017 14th Int. Bhurban Conf. Appl. Sci. Technol. IBCAST 2017*, pp. 328–332, 2017, doi: 10.1109/IBCAST.2017.7868073.

[21] P. Bertens, A. Guitart, P. P. Chen, and A. Perianez, "A Machine-Learning Item Recommendation System for Video Games," *IEEE Conf. Comput. Intell. Games, CIG*, 2018, doi: 10.1109/CIG.2018.8490456.

[22] A. Pathak, K. Gupta, and J. McAuley, "Generating and personalizing bundle recommendations on steam," *SIGIR 2017 - Proc. 40th Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, pp. 1073–1076, 2017, doi: 10.1145/3077136.3080724.

[23] R. Sifa, A. Drachen, and C. Bauckhage, "Large-scale cross-game player behavior analysis on steam," *Proc. 11th AAAI Conf. Artif. Intell. Interact. Digit. Entertain. AIIDE 2015*, vol. 2015-Novem, pp. 198–204, 2015.

[24] D. Parra and X. Amatriain, "Walk the Talk: Analyzing the relation between implicit and explicit feedback for preference elicitation," *Proc. 19th Int. Conf. User Model. Adapt. Pers.*, pp. 255–268, 2011.

[25] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan, "Beyond clicks: Dwell time for personalization," *RecSys 2014 - Proc. 8th ACM Conf. Recomm. Syst.*, pp. 113–120, 2014, doi: 10.1145/2645710.2645724.

[26] B. J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," *Res. J. Appl. Sci. Eng. Technol.*, vol. 5, no. 16, pp. 4168–4182, 2006, doi: 10.19026/rjaset.5.4644.

[27] J. Buder and C. Schwind, "Learning with personalized recommender systems: A psychological view," *Comput. Human Behav.*, vol. 28, no. 1, pp. 207–216, 2012, doi: 10.1016/j.chb.2011.09.002.

[28] A. Mittal and S. Subraveti, "Introduction to Recommender Systems Project Report Comparison of Recommendation Models On the Amazon Automotive Dataset," pp. 1–9, 2017.

[29] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer (Long. Beach. Calif).*, vol. 42, no. 8, pp. 30–37, 2009, doi: 10.1109/MC.2009.263.

[30] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-

factorization-based approach to collaborative filtering for recommender systems," *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014, doi: 10.1109/TII.2014.2308433.

[31] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *Proc. 2005 SIAM Int. Conf. Data Mining, SDM 2005*, pp. 471–475, 2005, doi: 10.1137/1.9781611972757.43.

[32] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 4–7, 2005, doi: 10.1109/ICDM.2005.14.

[33] N. Smar, J. A. Konstan, N. Borchers, J. Heriocker, B. Wer, and J. Wew, "Using Filtering Agents to Improve Prediction Qualityin the GroupLens Research Collaborative Filtering System," *World Wide Web Internet Web Inf. Syst.*, 1998.

[34] Y. Afoudi, M. Lazaar, and M. Al Achhab, "Collaborative filtering recommender system," in *Advances in Intelligent Systems and Computing*, 2019, vol. 915, pp. 332–345, doi: 10.1007/978-3-030-11928-7_30.